Hardening de servidores GNU/Linux

Carlos Álvarez Martín y Pablo González Pérez

Access Denied

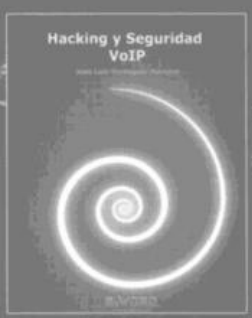
0xWORD
www.0xWORD.com

Mantener una infraestructura informática segura y al margen de atacantes siempre es una tarea complicada. Son muchos los aspectos a tener en cuenta. A los diferentes dispositivos en un entorno en red hay que añadirle la heterogeneidad de los sistemas operativos que la componen y los servicios prestados. Este libro trata sobre la securización de entornos Linux siguiendo el modelo de Defensa en Profundidad. Es decir, diferenciando la infraestructura en diferentes capas que deberán ser configuradas de forma adecuada, teniendo como principal objetivo la seguridad global que proporcionarán. Durante el transcurso de esta lectura se ofrecerán bases teóricas, ejemplos de configuración y funcionamiento, además de buenas prácticas para tratar de mantener un entorno lo más seguro posible. Sin lugar a dudas, los entornos basados en Linux ofrecen una gran flexibilidad y opciones, por lo que se ha optado por trabajar con las tecnologías más comunes y utilizadas. En definitiva, este libro se recomienda a todos aquellos que deseen reforzar conceptos, así como para los que necesiten una base desde la que partir a la hora de securizar un entorno Linux.

Carlos Álvarez Martín es un administrador de sistemas que obtuvo su titulación en el I.E.S. Gonzalo Nazareno. Ha trabajado en diferentes empresas dedicadas al software libre, con proyectos que abarcan desde el gestor documental Alfresco hasta el framework de comunicaciones Asterisk. De este último obtuvo la certificación dCap, Digium Certified Asterisk Professional, que le ha permitido colaborar en numerosos proyectos de comunicaciones IP. Posteriormente se unió al equipo de Informática 64, desempeñando las labores de docente y consultor de sistemas. Actualmente trabaja en Simosa IT, sociedad de Abengoa encargada de la definición y ejecución de la estrategia de informática y telecomunicaciones del grupo a nivel internacional, así como del soporte a usuarios y la operación de los sistemas.

Pablo González es Ingeniero en Informática por la Universidad Rey Juan Carlos de Madrid. Es Project Manager en Telefónica Digital. Ha trabajado como consultor de sistemas y seguridad en Informática 64, siendo responsable del departamento de seguridad. Dispone de certificaciones Microsoft, en el ámbito de Windows 7, servidores y virtualización, en calidad de MCTS y MCITP, y el LPIC-1 en Linux. Se le ha otorgado el Premio Extraordinario Fin de Carrera en Ingeniería Técnica en Informática de Sistemas en 2009. Premio al mejor expediente de su promoción por la Escuela Técnica Superior de Ingeniería Informática en 2009. Co-autor de otros libros con temática de seguridad como Pentesting con Kali o Hacking dispositivos iOS: iPhone & iPad.

Otros libros de **0xWORD**



Nivel: Avanzado - **Tipo de Libro:** Guía Profesional - **Temática:** Seguridad

0xWORD
www.0xWORD.com

Hardening de servidores GNU/Linux

Carlos Álvarez Martín

Pablo González Pérez

Índice

Prólogo **11**

Capítulo I. Introducción a la fortificación **13**

1. Introducción a la fortificación de entornos **13**

2. Defensa en profundidad **15**

Procedimientos, concienciación y políticas 16

Seguridad física 17

Seguridad del perímetro 17

Seguridad en la red interna 19

Seguridad a nivel de servidor 20

Seguridad en la aplicación 20

Seguridad a nivel de la información 21

3. Mínimo privilegio posible **22**

4. Mínimo punto de exposición **22**

5. Gestión de riesgos **24**

Capítulo II. Protección física **27**

1. BIOS / UEFI **27**

2. Gestor de arranque. GRUB y GRUB2 ..... **28**

Impacto de un gestor de arranque no protegido 28

Protección del gestor de arranque 31

3. Protección del sistema de ficheros **33**

Concepto de acceso a un sistema de ficheros 33

Cifrado de disco o particiones 34

4. Cifrado de ficheros **44**

Sobre GPG y su modo de funcionamiento 45
Cifrado simétrico con GPG 46
Cifrado asimétrico con GPG 47
5. Otras protecciones 49

Capítulo III. Protección perimetral 51
l.iptables 51
¿Qué es iptables? 51
Funcionamiento de iptables 51
Decisión de enrutamiento 52
Tablas 52
Agregando reglas con iptables 54
Listando reglas con iptables 56
Eliminando reglas aplicadas 56
Cambiando política por defecto 57
Haciendo las reglas permanentes 57
Firewall de '2 patas' 58
Firewall de '3 patas' 63
Front-ends para iptables 72
iptables e ipv6 73
2. VPN 75
Definición y tipos 75
PPTP, Point-to-point Tunneling Protocol 76
OpenVPN 80
3. Monitorización de la red 97
Icinga 97

Capítulo IV. Protección de la red interna 107
1. Spoofing o suplantación de identidad 107
ARP Poisoning 108
DHCP Spoofing 113
ICMP Redirect 116

2. VLAN 118
Configuración de VLAN en Linux 119
3. IPsec 120
Sobre el funcionamiento de IPsec 121
IPsec con Linux 125
4. IDS Snort 145
Instalación de Snort desde los repositorios oficiales de Debían 147

Capítulo V. Protección de la capa de aplicación 153
1. Jaulas con chroot 153
Prueba de concepto de una jaula con chroot 154
2. Permisos especiales, atributos y ACL 157
Un poco de teoría básica de permisos 157
Permisos especiales 158
Atributos 160
ACL, Access Control List 162
3. Elevación de privilegios con 'sudo' 166
Instalación de sudo y análisis de sus componentes 166
Ejemplo de configuración para sudo 170
4. Limitación de recursos 172
Inicio de sesión, passwords y límites 172
Cuotas de almacenamiento 182
Monit 185
5. Port-Knocking 188
SPA, Single Packet Authorization 189
6. Actualizaciones seguras en Debían 194
¿Es seguro apt? 194
7. HIDS, Host-based Intrusión Detection System 200
OSSEC 200

Capítulo VI. Fortificación de un entorno LAMP 211
1. Instalación de un entorno LAMP 211

2. MySQL 214
Dirección de escucha 214
Carga de ficheros locales 214
Renombrar el usuario root 215
Comprobar existencia de usuarios anónimos 215
Controlar los privilegios de los usuarios 216
mysqlsecureinstallation 216
3. PHP 216
expose_php 217
displayerrors 217
openbasedir 217
disablefunctions 218
Deshabilitar RFI 218
Suhosin 219
4. Apache 220
Configuraciones globales 220
Deshabilitar información ofrecida por el servidor 221
Configuraciones por contexto 222
modsecurity 226
HTTPS 228

Capítulo VII. Fortificación y seguridad en SSH 233
1. Introducción a SSH 233
Funcionamiento del protocolo 233
La primera conexión 235
2. Configuración del servicio 235
Archivos del servicio 236
Directivas básicas 238
Autenticación con contraseña 241
Clave pública y clave privada 242
Resumen del proceso de conexión 244
3. Aplicaciones con SSH 246

Copia segura con SCP 246
FTP seguro con SFTP 248
SSHFS: El sistema de archivos de SSH 248
XI1 forwarding con SSH 250
Fail2ban 250
4. Tunneling 253
SSH: tunneling 253
Túneles TCP/IP con port forwarding mediante SSH 256
5. SOCKS con SSH 256
Habilitando y utilizando SOCKS 257

Capítulo VIII. Logging 261
1. Consideraciones previas 261
2. rsyslogd 262
Clasificación de mensajes. Facility y severity 262
Configuración de rsyslogd 263
3. Rotación de logs 265
Ficheros de configuración de logrotate 265
Output channels y logrotate 268
4. Logging remoto o centralizado 269
Configuración de la máquina A 269
Configuración de la máquina B 269
Otras configuraciones interesantes 271

Índice alfabético 273

Prólogo

Hoy en día, cuando todo está informatizado y se depende en gran medida de la informática, resulta de vital importancia atender a todos los aspectos relacionados con la seguridad. Cada vez existen más amenazas que ponen en peligro la confidencialidad e integridad de los datos de empresas y particulares. En numerosas ocasiones se piensa que un sistema, servicios u otros recursos instalados resultan lo suficientemente seguros desde el primer momento como para implementarlos en producción. En el presente libro se estudiarán los métodos más comunes de fortificación para entornos basados en soluciones GNU ***/Linux.***

GNU ***/Linux*** es uno de los sistemas operativos más extendidos, principalmente en el ámbito empresarial actuando como servidores para diferentes tipos de roles. Su principal atractivo suele ser el alto grado de configuración y flexibilidad que ofrece a los administradores. Resulta muy habitual tener la sensación de que siempre se sabe qué se está haciendo en cada momento.

GN\J/Linux sin embargo es solo un componente, que unido a diferentes agrupaciones de software y configuraciones conforman las denominadas distribuciones. Cada distribución es diferente a la hora de instalar, mantener o modificar aplicaciones y servicios. Sin embargo todas disponen de un denominador común, que es la utilización de GNU ***/Linux*** como sistema base; además de la disponibilidad de grandes comunidades de usuarios, manuales y documentación que ayudan a sacar el máximo partido y rendimiento. En dichos manuales y documentación al alcance de cualquier persona, se incluyen configuraciones por defecto que facilitan y disminuyen el esfuerzo necesario para implementar una aplicación o servicio. El principal ***hándicap*** reside en que dichas configuraciones iniciales son públicas y por lo tanto inseguras a priori, por lo que deben ser modificadas y fortificadas.

Tradicionalmente las distribuciones implementadas en servidores han sido Red Hat y Debian GNU/***Linux***. En ambos casos la base es excelente para establecer los pilares de un sistema robusto, aunque no solo consiste en disponer de una buena base, sino en fortificar todos los elementos que la rodean. Para ello este libro sigue el esquema de ***Defensa en Profundidad***, que abarca aspectos comprendidos desde la seguridad física hasta la seguridad en las aplicaciones. Todo ello haciendo uso del sistema operativo Debian ***GN\J/Linux*** como base. A pesar de que todas las distribuciones se diferencian entre sí, la base y los conceptos que se tratan en este libro son aplicables a todas y cada una de ellas.

Este libro, por tanto, está dirigido a administradores de sistemas que deseen mejorar ciertos conceptos a la hora de fortificar un sistema, así como a personas que a pesar de tener experiencia en sistemas quieran iniciarse en el ámbito de la seguridad bajo sistemas GNU/***Linux.*** Como norma general, se tratarán de explicar conceptos y posteriormente aplicarlos a ejemplos prácticos con porciones de ficheros de configuración, esquemas de implementación, etcétera para que resulte más gráfico, práctico y ameno en su lectura.

Al finalizar el libro, el lector debe ser capaz de implementar las herramientas o complementos que se hayan expuesto, además de utilizar nuevas directivas de configuración en las herramientas o servicios, así como en el propio sistema operativo.

Capítulo I
Introducción a la fortificación

1. Introducción a la fortificación de entornos

Hoy en día la seguridad en las empresas es cada vez un aspecto más crítico en la gestión TI. El robo de información confidencial por parte de un usuario sin acceso a dichos datos, la denegación de un servicio, la suplantación de una identidad o la destrucción de información de la empresa son solo algunos riesgos a los que, día a día, un administrador se enfrenta.

Las organizaciones deben disponer de los siguientes elementos para disponer de un entorno seguro:

- Sistemas adecuados, tanto en su configuración como en la versión.
- Herramientas prácticas que realmente se utilicen con el fin de asegurar el entorno.
- Procedimientos de seguridad estrictos
- Conocimiento y capacidad

En un entorno corporativo se debe tener en cuenta los peligros a los que la empresa se encuentra expuesta. Uno de estos grandes peligros es el ***software*** y sus distintas vertientes, en función de lo que ejecuta.

El ***software*** fiable es aquel que hace lo que se supone que debe hacer, es decir, es una aplicación que ejecuta y realiza correctamente su tarea. El ***software*** seguro es aquel que hace lo que se supone que debe hacer y ninguna tarea más. Estos ***algo más*** son los que pueden producir inseguridad en un entorno corporativo.

Para estar seguro se debe ejecutar solo ***software*** perfecto, pero, este hecho es algo imposible ya que el ***software*** está realizado por los seres humanos, lo cual conlleva errores de programación. Otra de los procesos que se puede llevar a cabo para estar seguro es realizar procedimientos para mitigar ese ***algo más.***

La mejora de la seguridad en los sistemas TI es una de las máximas a la que se debe optar en un entorno corporativo. Es por ello que, habitualmente, se deben realizar procesos de fortificación de sistemas. Además, la ejecución de test de intrusión que comprueben hasta dónde se puede llegar y que se puede obtener. Los test de intrusión forman parte de las auditorías de seguridad informática.

Las auditorías se dividen en los siguientes tipos, en general:

- Caja blanca. La auditoría se realiza desde dentro de la empresa y se disponen de ciertos permisos de acceso a los sistemas. Se intenta verificar hasta dónde se puede llegar con ciertos privilegios y validar el estado de los sistemas y la configuración de estos.
- Caja negra. La auditoría se realiza desde el exterior. El atacante no dispone de ningún tipo de información más que los datos públicos de la empresa. Esta auditoría representa la visión de un ***hacker*** y se intenta verificar el estado de seguridad de los sistemas perimetrales y servicios públicos.
- Caja gris. La auditoría se realiza como si se fuese un empleado de la empresa y se intenta verificar hasta donde puede llegar un empleado descontento. No se dispone de gran privilegio, pero mediante técnicas para escalar privilegios se puede llegar a cierta información a la que, en principio, no se dispone acceso expreso.

Otra manera para mejorar la seguridad de los sistemas TI de una empresa u organización es mediante la formación de los técnicos que deben encargarse de la administración y gestión de los sistemas. Esta formación es uno de los puntos importantes en los empleados de una empresa, ya que con la rapidez que avanza la informática y la rama de la seguridad, es imprescindible disponer de unos empleados formados técnicamente.

La concienciación de los usuarios o empleados menos técnicos es algo también imprescindible. Los usuarios suelen ser víctima de numerosos engaños o ingeniería social, lo cual son un foco débil en la seguridad corporativa de la empresa. Se debe gastar recursos en concienciar a los empleados de la empresa para fortificar uno de los puntos débiles de la empresa.

La última medida que se aplicaría para fortalecer el entorno empresarial sería la aplicación de políticas adecuadas de seguridad y actualización de sistemas. Los sistemas desactualizados son otro de los puntos débiles de una empresa, un administrador descuidado puede provocar que los sistemas se encuentren en problemas. Las políticas y medidas de seguridad deben ser consensuadas y llevarlas a cabo cuando la seguridad de la empresa se encuentre comprometida.

El proceso de fortificación de sistemas debe ser un proceso automatizable en la mayoría de lo posible. Cierto es que la experiencia ayuda y mucho a la hora de proteger y fortificar los sistemas. Este proceso depende del rol que se esté tratando o funcionalidad instalada. Una recomendación es utilizar el mínimo de servicios por máquina, ya que una máquina con muchos servicios tiene una gran cantidad de focos que llaman la atención a usuarios malintencionados. Además el proceso de fortificación dispone de tres principios técnicos:

- Mínimo punto de exposición.
- Mínimo privilegio posible.
- Defensa en profundidad.

Las amenazas deben quedar claras en el escenario en el que se mueve el presente libro. Las amenazas intemas son, principalmente, los empleados descontentos, la fuga de información confidencial o la

seguridad física. Las amenazas externas son los ataques contra el perímetro de seguridad, los ataques ***client-side*** o ejecutados desde el lado del cliente, los ataques dirigidos contra algunas organizaciones o empresas y los diferentes tipos de ***malware.***

El presente libro organiza y explica estos conceptos en tomo a un modelo de defensa basado en la profundidad y unos principios técnicos. Siempre orientado a entornos ***GNU/Linux*** y la fortificación de estos y sus servicios.

GNU***ILinux*** proporciona una base muy utilizada en el mundo de los servidores y servicios públicos en Internet, pero se debe configurar y fortificar estos entornos para evitar problemas o sorpresas innecesarias.

2. Defensa en profundidad

El modelo de defensa en profundidad proviene del entorno militar, es decir, mantener múltiples líneas de defensa, en vez de disponer de una línea de defensa única muy reforzada. El fin de este modelo es el de retrasar el posible avance de un intruso o usuario malintencionado lo máximo posible.

En términos informáticos, este modelo propone la creación de capas de defensa con el objetivo de evitar un ataque directo a la información sensible de un entorno. Además, con este modelo se consigue un mayor tiempo para defenderse y utilizar planes de actuación y mayor probabilidad en la detección de un ataque.

Fig 1.01: Esquema del modelo *defense in depth* o defensa en profundidad.

Un símil interesante en la utilización de este modelo sería el de un banco, donde la información sensible sería el dinero que este dispone en la caja fuerte. Para proteger este dinero, se pueden

observar distintas medidas de seguridad como son barreras para evitar alunizajes, cámaras, guardias, un área pública dónde la gente espera su tumo, la cual podría ser la DMZ. Cuanto más cerca se encuentre de la caja fuerte, los controles de seguridad aumentan.

Es importante, en el instante de implementar este modelo, conocer de qué se dispone. En otras palabras, no es una buena práctica no disponer de un inventario y no conocer que tecnologías, soluciones de seguridad e incluso servidores se tienen en la empresa. Además, tampoco es una buena práctica disponer de tecnología por disponer, es decir, más vale tener un servicio, aplicativo, sistema operativo bien configurado que estos por defecto, simplemente por el hecho de tenerlo.

Otra de las características importante del modelo son las actualizaciones de ***software.*** Como se mencionó anteriormente, el ***software*** debe estar siempre actualizado. Es verdad, que históricamente las actualizaciones han provocado dolores de cabeza en los administradores. Hay que recalcar que antiguamente, las actualizaciones provocaban errores que podían hacer ***caer*** la estabilidad del sistema. Hoy en día, la programación de las aplicaciones es cada día mejor, aunque también más compleja, y la aplicación de parches no constituye un riesgo para la estabilidad de los sistemas.

La virtualización ayuda a generar entornos de pre-producción, los cuales son copias idénticas de los entornos de producción. Con este método se puede aplicar los parches a las distintas aplicaciones y comprobar que la estabilidad de los sistemas no se pierde. De este modo los administradores pueden estar seguros que al aplicar la actualización en un entorno de producción no habrá sorpresas.

Hoy en día, disponer de un entorno de pro-producción no supone un coste elevado y es asequible para casi cualquier empresa, el disponer de pocas máquinas con grandes recursos capaces de montar una copia de la infraestructura real. Esta infraestructura no debe disponer de una copia de la información sensible. Las características interesantes de la virtualización son el bajo coste y la existencia de las herramientas P2V, con las que de una máquina física se obtiene una instancia virtual.

A continuación se explican las distintas capas y los objetivos de seguridad que se buscan con su creación. A lo largo del libro se irán estudiando estas capas y la correcta configuración de ellas.

Procedimientos, concienciación y políticas

Los procedimientos son la mejor manera de llevar a cabo las tareas diarias. En el ámbito de la seguridad ocurre exactamente igual, es imprescindible automatizar y poder enumerar los pasos a seguir ante ciertas circunstancias que puedan ocurrir en el día a día de la empresa. Este hecho hace que la creación de procedimientos para la resolución de incidentes de seguridad es algo necesario en cada empresa. Estos procedimientos dependerán de la estructura y organización de la empresa.

La concienciación es algo necesario para los empleados. Como se mencionó anteriormente, los usuarios no técnicos, e incluso algunos técnicos, no ven peligros en el uso de cierta información o ciertos sistemas. Es por ello, que nacen otras medidas como la del mínimo privilegio posible, para contrarrestar el efecto negativo que pueden suponer las acciones de ciertos usuarios en algunos sistemas.

¿Cómo llevar a cabo la concienciación? Esto es algo complejo, lo óptimo es la utilización de cursos en la que los usuarios vean lo fácil que puede llegar a ser el acceso a ciertas partes de su información privada o sensible. Entonces se puede decir que la formación a usuarios es algo vital, pero también siendo realistas se sabe que existirán usuarios que seguirán utilizando malos hábitos, como puede ser el ***post-it*** con su contraseña pegado en la pantalla de su equipo.

Las políticas de seguridad son algo también de vital importancia. Como política de seguridad, encontrándose en el escalón más bajo en la imagen anterior, se entiende a las acciones con las que se obligan a ciertos usuarios a actuar. En otras palabras, es importante que algunos usuarios, que disponen de ciertos roles utilicen unas políticas de actuación y hábitos.

Lo ideal, sería aplicar estar políticas de seguridad a todos los usuarios de la empresa, pero es algo complejo, ya que cada uno dispone de una relevancia distinta y acceden a distintos tipos de información. Ya que, generalmente, estas políticas no son aplicables a todos los usuarios, se dispone de otras medidas como son las de mínimo privilegio posible y mínimo punto de exposición que ayudan a contrarrestar efectos negativos o malos hábitos de los usuarios.

Seguridad física

En este eslabón del modelo de defensa en profundad se puede entender dos aspectos muy distintos. En primer lugar, se puede visualizar la seguridad física como el procedimiento mediante el uso de cámaras, guardias de seguridad, CPDs aislados y asegurados que protegen las distintas capas o el acceso a sus contenedores. En segundo lugar, la seguridad o protección física se podría entender como los mecanismos que son utilizados para asegurar los sistemas o la información del acceso físico a un medio digital por parte de un usuario.

La utilización de sistemas de vigilancia, cámaras, guardas que protejan los datos de la empresa, por ejemplo en la noche, es de vital importancia. Pero en el presente libro lo que compete es la segunda visión de seguridad física. La utilización de mecanismos que eviten que un usuario con acceso físico a los equipos pueda realizar cualquier tarea con ellos.

Hay que recalcar que cuando un usuario dispone de acceso físico a un equipo, en la mayoría de los casos, será difícil evitar que pueda utilizar el sistema para realizar tareas de dudosa moral. Más adelante en el libro se podrá estudiar mecanismos para evitar que un usuario pueda arrancar un sistema operativo si él no está autorizado, que pueda llegar a ser ***root*** con suma facilidad ó proteger la información sensible mediante el uso del cifrado.

Seguridad del perímetro

El perímetro es una de esas barreras dedicadas a proteger el entorno o capa interna de la empresa. Es el paso previo a la red interna, y es una capa que debe estar correctamente configurada y conocer profundamente.

El ***firewall*** es el mayor representante de esta capa de seguridad. Representan un mecanismo de defensa inicial y que esta compuesto por reglas. Una buena práctica sería un ***firewall*** lo más restrictivo posible, estableciendo las reglas por equipo y servicio. En muchas ocasiones, estos dispositivos disponen de gran cantidad de posibilidades desconocidas por el encargado de su configuración, lo cual puede provocar que exista tráfico, por ejemplo mal formado, que pueda colarse entre esta barrera. Es por ello, que el conocimiento es necesario para una correcta configuración de dichos dispositivos. En el presente libro se tratará la extensa configuración de ***iptables*** en un entorno GNU/ ***Linux*** para proteger la organización.

Las listas de control de acceso ó ACL de los ***routers*** son otra de las piezas claves en esta capa. La configuración de este tipo de mecanismo ayuda a evitar que el tráfico no deseado no llegue a lugares dónde no está planeado.

Hay que asegurarse que tras la configuración de ambos mecanismos los datos provenientes de equipos permitidos siguen fluyendo a lo largo de la red. En algunas ocasiones se pueden generar ataques de denegación de servicio a la propia empresa por una mala configuración de un ***firewall*** o una ACL, lo cual hará que el encargado de ello sea llamado rápidamente ante una posible queja.

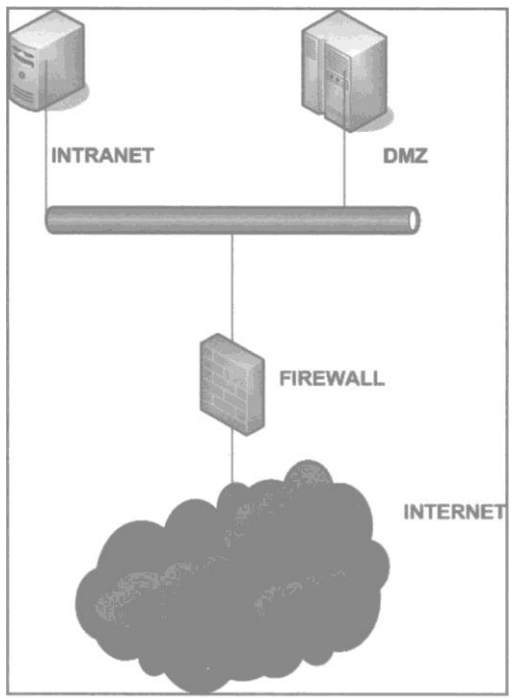


Fig 1.02: Ejemplo esquema de *firewall*.

Otro mecanismo que se debe implantar son las soluciones ***Virtual Prívate NetWork***, también conocidas como VPN. La VPN debe utilizarse entre el perímetro de la red y los segmentos internos

que necesiten más seguridad. Además, mediante el uso de una VPN se puede asegurar los protocolos sin cifrado en ambientes que dicho tráfico puede caer en manos de terceros. Para los usuarios remotos, se debe utilizar una VPN que requiera de autenticación de factores múltiples y que utilice una tecnología como IPSEC, SSL o SSH para crear el túnel a los recursos de la red corporativa o interna utilizando reglas de control de acceso por usuario. Una recomendación es la utilización de auditorías, cada cierto tiempo, de acceso de los usuarios para el dispositivo VPN.

NetWork Access Protection o NAP es otro mecanismo que se puede estudiar y configurar para dotar de seguridad a esta capa. NAP evita el acceso a la red corporativa de los equipos si no cumplen con unos requisitos previos. Se puede dejar en cuarentena a un equipo, e incluso, conectarle a una red auxiliar dónde pueda disponer de Internet, pero no de conectividad con equipos de la red corporativa.

802.IX es otra de las tecnologías que se puede utilizar para autenticar, por ejemplo, un equipo en la red. También entraría en esta capa, como protección perimetral.

La separación entre la zona desmilitarizada ó DMZ y la red interna es una obligación. El ***firewall*** se encargará de gestionar el tráfico entre Internet, la/s DMZ y la red interna.

Seguridad en la red interna

La capa que fortifica directamente la red interna de la organización dispone de distintos mecanismos, los cuales suman y aportan su granito de arena. En el presente libro se estudia la segmentación de redes, las ***virtual local area network*** o VLAN y los IDS o ***intrusión detection system.***

Se debe utilizar la segmentación para separar las redes en función de los usuarios que quieran acceder y, en función, de los recursos que se requieran. Es importante evitar que, por ejemplo, un cliente que llegue a la empresa se pueda conectar directamente a redes corporativas a las que no se desea que un usuario externa acceda.

El nivel de enlace es una de las capas de la arquitectura OSI, la cual dispone de un ataque básico pero muy eficiente, como es ***man in the middle.*** Desde el punto de vista de la seguridad, el aislamiento de las LAN a través del uso de VLAN es una buena práctica para evitar que ciertos usuarios de la misma red puedan acceder a ciertos equipos a los que no deban disponer de conectividad. Con las VLAN se fortalece el nivel de enlace, el cual en muchas ocasiones es un punto débil en las empresas. Hay que tener en cuenta que en una auditoría de caja gris o blanca, el nivel de enlace puede suponer un amplio campo de batalla por el que usuarios malintencionados puedan interceptar comunicaciones e información.

Los IDS son sistemas casi obligatorios hoy en día, al menos muy recomendada su implementación en una organización. Con este tipo de soluciones se puede detectar accesos no autorizados a un equipo, o incluso a una red. Estos ataques pueden ser el resultado de ***hackers*** o herramientas automáticas. Los IDS están, generalmente, compuestos por un ***sniffer*** que capta el tráfico de una red y tras realizar un análisis se puede llegar a indicios de posibles ataques. Hay que tener cuidado con los falsos positivos que también pueden surgir de la implantación de un IDS y su análisis del tráfico de una red.

El IDS no solo analiza qué tipo de tráfico es, sino que también analiza el contenido y su comportamiento. Hoy en día, la integración del IDS con el ***firewall*** ayuda, en gran medida, a fortificar la capa de seguridad dedicada a la Intranet. Los IDS disponen de una gran base de datos con las firmas de los ataques conocidos, pero también son capaces de analizar el tráfico en busca de contenido sospechoso o comportamientos extraños.

Existen distintos tipos de IDS, como pueden ser los HIDS, ***host intrusión detection system,*** y NIDS, ***network intrusión detection system.*** El HIDS depende del éxito de la intrusión por parte del usuario malintencionado, normalmente, estos dejan un rastro de las actividades que han llevado a cabo en el equipo vulnerado. El HIDS intenta detectar estas actividades o rastro dejado en las modificaciones de los equipos afectados y realizar un reporte con las conclusiones. Por otro lado un NIDS

Seguridad a nivel de servidor

La seguridad a nivel de servidor puede ser entendida de distintas maneras, sin llegar a interceptar a la seguridad del nivel de aplicación. En esta capa se puede tener en cuenta las actualizaciones del sistema operativo servidor que da soporte y gestiona las aplicaciones y servicios que se ejecutan en dicha máquina.

Es importante disponer de logging, tanto local como remoto, en la máquina servidor para llevar un registro tanto de actividad, pudiendo realizar en cualquier instante un proceso forense para detectar que está ocurriendo en el servidor.

Existen distintos tipos de logs en el sistema como pueden ser los de kemel, autenticación, registro de actividad o tareas, etcétera. Saber gestionar esta información es algo realmente útil para el administrador que se preocupa de la seguridad de sus sistemas.

Las actualizaciones en los sistemas son un hecho prácticamente diario, ya que cada día salen un gran número de vulnerabilidades, tanto en aplicaciones como sistemas operativos. Por esta razón, hay que disponer de un plan de contingencia para estar preparado para cubrir y mitigar estas vulnerabilidades.

Uno de los objetivos de la defensa en profundidad en la capa del servidor es dejar a este en ***Zero Day Server.*** En otras palabras, este estado proporciona a la empresa y al administrador encargado de la gestión de los servidores un sistema sin vulnerabilidades conocidas. Este es el máximo nivel de seguridad que se puede alcanzar con el software que se ejecuta en un sistema. Este estado puede ser puesto en duda con la aparición de ***Zero Day Exploits.***

Seguridad en la aplicación

La seguridad a nivel de aplicación es uno de los puntos importantes a tratar, si las capas que soportan a esta están bien cubiertas. Generalmente se dispone de varias aplicaciones o servicios que pueden tratar con la parte pública a través de la red, y la exposición de esta debe ser controlada y segura.

Una configuración por defecto puede proporcionar vías de ataque a un usuario malintencionado con las que lograr acceder al control remoto de la máquina, una denegación de servicio o simplemente a visualizar la configuración interna de la máquina. Por ello es mejor disponer de una configuración propia y en la que se sepa que se está realizando a disponer de un gran número de aplicaciones configuradas por defecto, las cuales pueden poner en riesgo la seguridad de la organización.

Existen usuarios malintencionados que tienen acceso a los recursos de la empresa, aunque siempre se tiende a pensar que los compañeros no realizarán acciones maliciosas, puede ocurrir que sí. Por esta razón se debe disponer de las aplicaciones actualizadas, para evitar un posible proceso de explotación y elevación de privilegios. Además, y como se ha comentado anteriormente, se debe controlar que usuarios pueden realizar acciones con privilegios. No es una buena práctica disponer de un usuario ***root***, el cual conozcan distintas personas. Se debe utilizar una política ***sudoer***, limitando el uso de los usuarios en sus acciones críticas. Además con los ***sudoer*** se cumple el principio de mínimo privilegio posible, el cual es una de las máximas que deben cumplir los sistemas para aumentar la seguridad.

Las cuotas de disco también puede favorecer que los usuarios no se excedan con los recursos, por ello debe haber un control de almacenamiento para cada usuario.

Por último comentar que la fortificación de los servicios mediante políticas de máxima restricción es deseable en los sistemas. Servicios como Apache, SSH, entre otros, deben ser lo más restrictivos posible para proporcionar un ambiente seguro de interacción con el usuario y favorecer la impresión de seguridad en la organización. Este tipo de aplicación debe ser ejecutada con usuarios ***no root***, cuando sea posible. Las jaulas, proporcionan al administrador un escenario donde los usuarios no podrán subir por encima de un cierto nivel de directorios, dando acceso a los recursos de dichas rutas, pero no a otras.

Seguridad a nivel de la información

Uno de los mayores activos que dispone una organización son los datos e información de esta. Estos activos deben estar bien protegidos, tanto por las capas inferiores del modelo Defensa en Profundidad, como por la propia capa de seguridad.

El cifrado de información es algo totalmente necesario hoy en día, las organizaciones o empresas manejan información sensible y esta debe ir protegida por algoritmos de cifrado robustos. Un simple envío de correo electrónico puede poner en jaque la seguridad de la empresa desvelando información sensible, la cual nunca se sabe dónde puede acabar. En este libro se podrá estudiar distintas vías para proteger la información gracias al cifrado de esta.

Los servidores pueden requerir de particiones cifradas con las que poder evitar el arranque de otro sistema y poder visualizar información sensible. Es importante que todas estas acciones queden registradas mediante logs de información.

3. Mínimo privilegio posible

Es muy común visualizar como los usuarios que ejecutan aplicaciones en una máquina lo hacen, en una gran mayoría de las ocasiones, con los máximos privilegios posibles. Los usuarios de sistemas Windows, por naturaleza, han utilizado siempre al usuario administrador o a un usuario normal que también pertenecía al grupo de administradores. Este hecho pone en riesgo al sistema, ya que si el usuario ejecuta cualquier binario malicioso este correrá en el sistema con los máximos privilegios. Lo mismo ocurre en un sistema GNU ***/Linux,*** las aplicaciones deben ejecutarse con el mínimo privilegio posible, para evitar que si esta es vulnerada o si se ejecuta una aplicación maliciosa se dañe el sistema.

La recomendación es que los usuarios trabajen sin privilegios y que solo se introduzcan estos en el instante que se requiera ejecutar una instrucción con elevación. En los servidores GNU/Lmwjc se dispone de los ***sudoers***, con los que se puede implementar estas acciones. No es recomendable que un usuario ***root*** sea conocido por distintos empleados de la organización. La acción correcta es que haya varios usuarios que puedan ejecutar acciones como ***root***, y esto se encuentre definido y configurado en el archivo ***sudoers.*** Además, se puede indicar que acciones podrán realizar con la identidad de ***root*** dichos usuarios, o cuáles no. Realmente es muy interesante la configuración de los ***sudoers*** para proteger el servidor de accesos o ejecución de instrucciones ilícitas.

Cuando se habla de mínimos privilegios posibles, no solo se trata el tema de la ejecución de procesos, sino que también se determina los privilegios de los directorios. Es importante recalcar que los directorios deben estar correctamente configurados por los usuarios, dando permisos a quien crean conveniente. Es importante que el administrador gestione los permisos, no solo los básicos de los sistemas ***Linux***, sino también los permisos avanzados.

Los bits especiales también pueden ayudar a proteger los sistemas ante borrados inapropiados, ***sticky bit***, o la ejecución de ciertos comandos con la identidad del propietario, por ejemplo del bit ***SUID.*** Es importante conocer toda la arquitectura y mecanismos, tanto básicos como avanzados, que proporciona un sistema GNU ***/Linux,*** y de este modo disponer de un entorno lo más fortificado posible.

Es realmente común encontrarse con instalaciones por defecto, y como se mencionó anteriormente no son seguras debido a que estas configuraciones por defecto pueden implicar que dicho proceso sea ejecutado por un usuario con privilegios. Si un usuario malintencionado vulnerase dicho proceso obtendría los privilegios del usuario que lo ejecutó.

4. Mínimo punto de exposición

El mínimo punto de exposición es difícil de marcar a priori. Pero suponga el siguiente escenario: se dispone de varios servidores que tienen alguna vulnerabilidad conocida y los cuales no tienen una actualización que solvente dichas vulnerabilidades. Cuanto menor sea el punto de exposición

menos probabilidad de que ataquen dicha máquina, aunque el riesgo existe. Por esta razón se debe lanzar un plan de contingencia, el cual mitigue el efecto de la amenaza. Cuanto menor sea el punto de exposición, menor será la amenaza que hay que mitigar o el impacto de esta.

Un sistema debe ejecutar aquel software que sea imprescindible para el correcto desempeño de sus funciones. Hay que realizar un análisis de los servicios que se ejecutan en un servidor e ir indicando cuales son críticos, cuales medios y cuales bajos. En un servidor donde existe un servicio crítico no se debe compartir otros servicios, ya que la vulneración de uno de estos puede suponer el acceso a los datos que gestiona el otro servicio. Es por esta razón que no es recomendable ejecutar varios servicios sobre las mismas máquinas. Una posible ayuda a este hecho es la utilización de la virtualización para aislar máquinas dentro de un mismo servidor físico.

Otro punto a tener en cuenta es el olvido de servidores antiguos, de pruebas, de desarrollo, etcétera. Este tipo de acciones pueden poner en riesgo la seguridad de la organización. Además, estas acciones suelen ir acompañadas de procesos con privilegios y contraseñas débiles o sencillas ya que se utilizaban en un entorno de pre producción. Al situar este servidor o servicio de prueba en el mapa de la producción se crea un agujero de seguridad que hay que tener en cuenta y mitigar en caso de ser detectado. Las instalaciones por defecto con alta superficie de exposición también deben ser controladas.

Para ayudar y mitigar la exposición de los servicios o máquinas se puede utilizar la segmentación de redes. Esta solución se utiliza en el modelo en Defensa en Profundidad para asegurar la red interna y poder delimitar el tráfico que circula por ella. La segmentación de redes ayuda y mitiga el impacto de ciertos ataques a la red interna, tanto en el nivel de enlace como en los niveles superiores de la arquitectura OSI, ***Open System Connection.***

La DMZ, zona desmilitarizada, es el punto en el que se encuentran los servicios públicos de la empresa. Esta zona debe estar controlada y monitorizada, y disponer de elementos, tanto lógicos como físicos que detecten actividad sospechosa. Es una zona creada para que los usuarios externos accedan a los recursos públicos de la empresa de una manera controlada.

La configuración de las empresas puede variar en función de los fondos económicos dedicados a la gestión de la seguridad. Generalmente, se dispone de varias configuraciones como pueden ser:

- Configuración de tres patas. En esta configuración se dispone de un***firewall*** el cual conecta la entrada pública con la DMZ, y también se dispone de una salida con la parte pública. Por eso se denomina tres patas, ya que el mismo ***firewall*** gestiona las tres zonas.
- ***Front-End y Back-End.*** En esta configuración se dispone de dos ***firewall***, el primero es el denominado ***Front-End***, con el que se gestiona Internet y la DMZ. El segundo ***firewall*** denominado ***Back-End*** gestiona la unión entre la DMZ y la zona interna. Esta configuración necesita mayor recursos económicos pero está considera más segura, siempre y cuando su configuración sea la correcta.
- ***Front-End*** y ***Back-End*** con alta disponibilidad. Esta configuración es similar a la anterior, salvo que dispone de dos salidas a Internet para asegurar la conectividad y alta

disponibilidad. Además, se disponen de dos ***firewall*** en el ***Front-End*** y otros dos en el ***Back-End.*** Es recomendable que sean de dos fabricantes distintos para evitar que si se vulnera uno, se acceda a la otra zona por inercia. Además, las conexiones de los ***firewall*** deberían encontrarse cruzadas, es decir, un fabricante conecta con el otro. Esta configuración se puede encontrar o heredar del Esquema Nacional de Seguridad.

5. Gestión de riesgos

La seguridad es la capacidad de las redes o de los sistemas de información para resistir, proporcionando un nivel suficiente de confianza, los accidentes o acciones maliciosas que pueden comprometer la disponibilidad, autenticidad, integridad y confidencialidad de los datos almacenados y los servicios que se ofrecen. Es realmente importante mantener estos cuatro principios siempre con un nivel aceptable.

El objetivo de todo lo que se verá en el libro y de todas las metodologías de gestión de riesgos que se pueden implantar en una organización es la de proteger la misión de la empresa. Para proteger la misión u objetivo de la empresa se deben tener en cuenta las distintas dimensiones de seguridad y analizarlas una a una:

- Disponibilidad. Disposición de los servicios a ser usados cuando se requiera de ellos. La disponibilidad afecta directamente a la productividad de las organizaciones.
- Integridad. Esta dimensión afecta a la completitud o corrección de los datos o información que es gestionada por la empresa. La integridad afecta directamente al correcto desempeño de las funciones de una empresa.
- Confidencialidad. La información debe ser accedida solo por personas autorizadas para ello. Es una de las dimensiones importantes, ya que puede minar la confianza de los usuarios o clientes sobre la imagen de la empresa.
- Autenticidad. Que no exista duda de quién se hace responsable de una información o servicio. Los engaños o suplantaciones son los peligros a los que se enfrenta esta dimensión de la seguridad.

En un proceso de gestión de riesgos se debe proponer un nivel de riesgo aceptable por la empresa. Existe un riesgo intrínseco y otro residual. El primero es el riesgo y el impacto de este sobre los activos de la organización sino se tuviera en cuenta las medidas de seguridad que protegen a los activos. El riesgo residual es el valor del riesgo teniendo en cuenta las medidas de seguridad que se implantan en la organización para salvaguardar los activos.

En un proceso de gestión de riesgos se siguen los siguientes pasos:

1. Identificación de activos. Este primer paso es uno de los más importantes ya que se intentan determinar los activos más relevantes para la organización, su interrelación y su valor. Además, es importante asignar el perjuicio o coste que supondría su degradación para la organización.

Los activos, generalmente, se jerarquizan en capas. Los activos más importantes para la empresa pueden ser la información relevante a esta, la imagen corporativa, la integridad de sus empleados, pero para que estos activos estén protegidos existen otros activos en capas inferiores, como pueden ser un cortafuego, guarda de seguridad, cifrado de datos. Los activos inferiores pueden adquirir un valor mayor debido a que heredan la importancia en función de a quién protejan.

2. Determinación de amenazas. Hay que realizar un listado de las amenazas que pueden afectar a cada activo de los que dispone la empresa. En la identificación de activos es importante que participe toda la empresa, entre distintos departamentos y que se consiga un punto unánime entre todos. Las amenazas hay que describirlas, indicando a que dimensión de seguridad afecta, pudiendo afectar a más de una.
3. Determinar las medidas de protección. Hay que realizar un listado de las medidas de protección que se disponen para proteger los activos y cuán eficaces son frente a los riesgos a los que se enfrenta la empresa.
4. Estimación del impacto. Se debe realizar una estimación del impacto entendiendo a este como el daño sobre el activo derivado de la materialización de la amenaza. En este paso se entiende que no se tienen medidas de protección puestas en producción.
5. Estimación del riesgo residual. Se debe realizar una estimación del impacto con las medidas de protección configuradas.

El riesgo residual y el riesgo intrínseco deben compararse para ver si la medida de protección es efectiva o no. La dirección deberá aprobar el riesgo residual, siempre y cuando la consideren un nivel de riesgo aceptable para la organización.

Capítulo II
Protección física

En este punto se van a exponer una serie de técnicas para tratar de mitigar o al menos retrasar el acceso a un servidor estando físicamente en el emplazamiento del mismo. Se está suponiendo que el atacante ha saltado la primera capa de la protección física; es decir, ha eludido cámaras de vigilancia, guardas de seguridad, cerraduras electrónicas, sensores de presencia, etc. Ahora el atacante está ante varios racks que albergan numerosos servidores físicos y tiene acceso físico a las máquinas. De igual modo, está equipado con herramientas para hardware y software que van desde un simple destornillador de tipo ***thor*** que le ayudará a extraer un disco duro, hasta dispositivos USB o CD auto- arrancables con herramientas de extracción de datos, cracking de passwords, etc.

Se exponen a continuación algunas recomendaciones para ponerle las cosas difíciles a este atacante.

l. BIOS / UEFI

Lo primero que se debe hacer en un equipo una vez que se configura y forma parte de un entorno de producción consiste fortalecer la configuración básica del BIOS, o su reciente sustitución y modernización para sistemas con hardware moderno llamada UEFI.

La fortificación de esta capa es un punto muy relativo y depende del fabricante del hardware, aunque generalmente los sistemas BIOS / UEFI se asemejan entre sí.

Los puntos que habría que reforzar en todos los entornos serían los listados a continuación, obviamente sin dejar de incluir aquellas soluciones específicas de cada fabricante o modelo de hardware:

- Edición de opciones protegida por contraseña.
- Deshabilitar la selección de medio de arranque.
- Deshabilitar, si fuera posible, los siguientes medios de arranque.
  - Cualquier medio de tipo extraíble. Por ejemplo los dispositivos USB.
  - Arranque desde todas las tarjetas de red del sistema usando sistemas PXE.

Si el atacante consiguiera acceso a puertos de periféricos externos, unidades ópticas, tarjetas de red, etc. se encontraría con una barrera en caso de haber deshabilitado el arranque desde los medios

mencionados. Debe tenerse en cuenta que no siempre es posible deshabilitar todos los medios de arranque y en el caso del arranque por red hay que ser cauto para no deshabilitar el inicio de la máquina por red mediante la opción ***Wake on Lan.***

Lo próximo que intentaría el atacante sería el acceso al BIOS o la UEFI para tratar de modificar el comportamiento de arranque de la máquina. Estableciendo una contraseña para la edición de opciones se erradicaría el problema. Es recomendado por otro lado establecer una contraseña si fuera posible para los propios configuradores y BIOS exclusivos para los RAID, evitando así la manipulación de la configuración de los discos físicos.

2. Gestor de arranque. GRUB y GRUB2

Obviamente, todas las medidas de protección mencionadas anteriormente son muy débiles cuando el atacante está frente al servidor y dispone de acceso al hardware, retirar la pila que mantiene salvada la configuración del BIOS, retirar los discos, etc. Pero antes de "perder" tiempo accediendo al hardware, el atacante se dispone a ver qué sistema operativo ejecuta la máquina física haciendo uso del algún monitor situado en el CPD.

Cuando se arranca un sistema y aparece el gestor de arranque GRUB, o su versión más moderna GRUB 2, se sabe de antemano que se está ante una posibilidad bastante probable de saltar las credenciales del servidor. Como consecuencia de ello, disponer de acceso a la visualización de datos sensibles, así como extraerlos para su posterior estudio o explotación. La extracción puede realizarse en soportes físicos insertados localmente en la máquina atacada, o bien volcándolos a servidores remotos haciendo uso de los comandos SCP o FTP por ejemplo.

Impacto de un gestor de arranque no protegido

Dándose el caso anterior y disponiendo de un teclado y un monitor, se pueden realizar las siguientes acciones independientemente de la versión de GRUB.

- Ejecución de comandos GNU básicos.
- Acceso a una Shell como root.
- Arranque de un sistema editando las líneas de configuración.

En el caso de GRUB2, al iniciar la máquina aparece una pantalla como la siguiente donde a priori solo es necesaria para seleccionar qué sistema operativo y/o kernel se desea ejecutar.

Como se observa en la descripción de funcionamiento, en la zona inferior de GRUB, es posible editar las líneas que permitirán arrancar el sistema operativo, además de poder acceder a una terminal que proporciona el mismo GRUB. Hay que recordar que aún no está arrancado el sistema operativo y que hasta el momento no se ha solicitado ningún tipo de credencial.

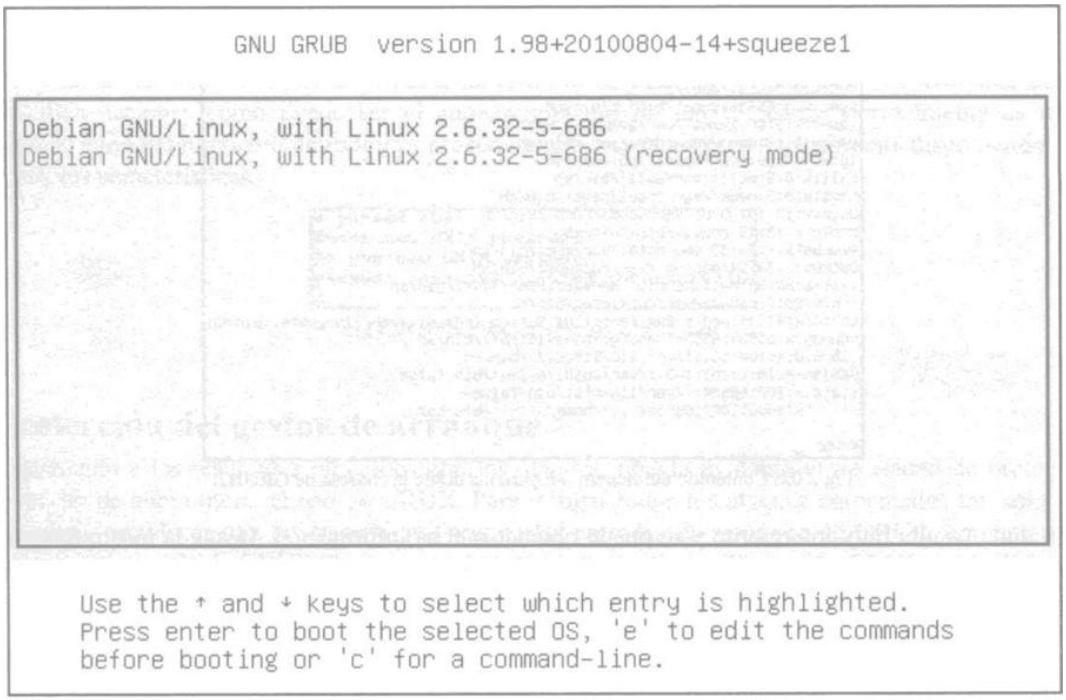


Fig 2.01: Pantalla principal del gestor de arranque GRUB2.

Utilizando la terminal de GRUB

El primero de los ataques consiste en visualizar información sensible de la máquina atacada usando la línea de comandos de GRUB. En el caso de GRUB2 se pulsa la tecla "C". Aparecerá una consola con una breve ayuda para su uso y el ***prompt*** grub>.

Pulsando sobre la tecla ***Tab*** o bien escribiendo la palabra reservada ***help,*** aparecerá un listado con los comandos disponibles. De todos los comandos mostrados, son interesantes ***Is*** y ***cat.*** Con ellos se pueden volcar listados de ficheros y sus contenidos, como ocurre en las consolas tradicionales una vez arrancado un sistema ***Linux.***

Un par de ficheros interesantes pueden ser ***passwd*** y ***shadow.*** Pero, ¿cómo se puede visualizar el contenido de dichos ficheros? ¿Cuál es su ruta? - La ruta puede averiguarse escribiendo en la consola el comando ***Is*** -/, con el que se podrá obtener qué partición o disco es el objetivo. Se supone que la partición del sistema, contenedora de los ficheros anteriores, es ***(hdO,msdosl).*** En tal caso se podría volcar el contenido de los ficheros escribiendo ***cat (hdO,msdosl)/etc/passwd.*** La salida de dicho comando mostraría el contenido del fichero ***/etc/passwd*** del sistema víctima. Como se puede comprobar, desde este entorno no hay permisos que valgan. No importa que un fichero posea permisos 600 y ***root*** como propietario del mismo.

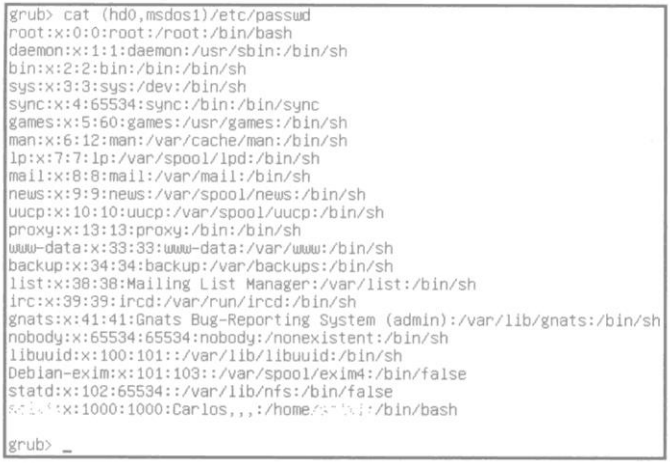


Fig 2.02: Contenido del fichero */etc/passwd* desde la consola de GRUB2.

Sin duda resulta muy interesante y se puede obtener mucha información. Desde la mostrada hasta ficheros personales, Scripts, reglas de ***firewall***, configuraciones, credenciales en claro de conexiones a bases de datos en aplicaciones web, usuarios, servicios, etc. Aunque todo ello puede resultar poco útil mostrándolo en una pantalla sin posibilidad de extraerlo.

Obteniendo acceso root desde la terminal de GRUB

Es el tumo de utilizar GRUB para obtener acceso como ***root*** sin conocer las credenciales. Consiste en editar la línea deseada para que, en lugar de realizar un arranque convencional, se ejecute una ***Shell*** con privilegios de súper-usuario. Para editar las líneas de arranque, estando en el menú principal de GRUB y situado sobre la línea deseada, es necesario pulsar la tecla "E". Dicha acción abrirá una edición de la entrada de GRUB. Se debe arrancar el núcleo ***Linux*** en modo escritura, mono-usuario y ejecutar una terminal sh. Las opciones anteriores son "rw single init=/bin/sh".

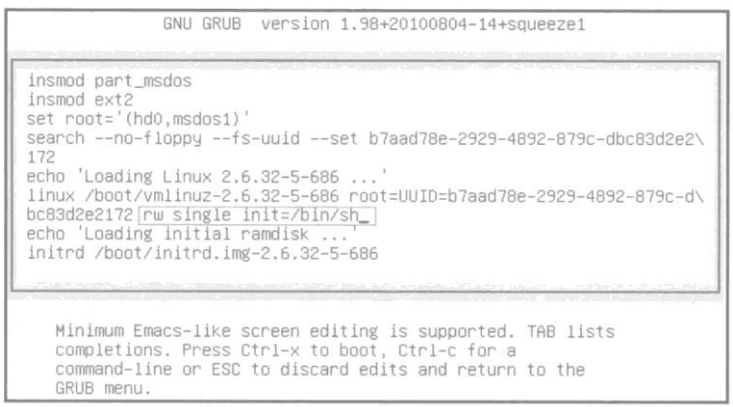


Fig 2.03: Edición de la entrada de arranque para obtener una *Shell* como *root*.

Una vez que el núcleo arranque aparecerá el símbolo #, que indicará que se tiene a disposición un ***prompt*** como súper-usuario, con las particiones montadas y con la posibilidad de realizar modificaciones sobre el sistema. A pesar de todo, la terminal no resulta del todo cómoda, pues le faltan acciones como puede ser el autocompletado; lo que se realiza normalmente es una modificación del ***password*** de ***root*** y se procede a reiniciar el sistema normalmente disponiendo de todas sus características.

```
# passwd root
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
#
```

Fig 2.04: Modificando el *password* de *root.*

Protección del gestor de arranque

Atendiendo a los resultados obtenidos con los ataques, resulta evidente la necesidad de proteger o limitar de algún modo el uso de GRUB. Para mitigar todos los ataques comentados tan solo es necesario establecer una contraseña que proteja las entradas de arranque de GRUB, así como el acceso a la consola de GRUB. Éste es uno de los puntos donde se diferencian GRUB y GRUB2.

Protección de GRUB

En la primera versión del gestor de arranque se puede establecer una especie de contraseña maestra que ayudará a proteger la edición y el acceso a la consola de GRUB. Esto resulta suficiente para evitar los ataques mostrados anteriormente.

La contraseña puede establecerse en el fichero de configuración de GRUB en texto plano o bien estando cifrada. Obviamente se recomienda el uso de una contraseña cifrada, ya que en caso de visualización de la configuración de GRUB por alguien no deseado, resultará más complejo obtener la contraseña. En el caso de Debian, el fichero de configuración de GRUB es ***/etc/grub/menu.lst.*** Lo único que habría que agregar al fichero de configuración es la palabra reservada***password*** seguida de una contraseña en texto plano, o bien la expresión ***password —md5*** seguida de un hash MD5. Dicha directiva debe establecerse al inicio del directorio y quedaría del siguiente modo.

```
timeout         5

# Pretty colours
color cyan/blue white/blue

## password ['--md5'] passwd
# If used in the first section of a menu file, disable all interactive editing
# control (menu entry editor and command-line)  and entries protected by the
# command 'lock'
# e.g. password topsecret
#      password --md5 $1$gLhU0/$aW78kHK1QfV3P2b2znUoe/
# password topsecret
password --md5 $1$2YB2x0$quuW3VKro2FiYyNo639Ew0
```

Fig 2.05: Estableciendo un *password* en GRUB.

Para obtener el ***password*** en formato ***md5*** con ***salí*** se utiliza la herramienta ***grub-md5-crypt.*** Otra opción interesante a la hora de proteger GRUB utilizando el ***password*** establecido, consiste en establecer el comando ***lock*** en cada una de las entradas de arranque que se quieran bloquear. Esto último evitará la posibilidad de arrancar una o varias entradas determinadas si no se ha introducido el ***password*** de bloqueo.

Una vez reiniciada la máquina no será posible acceder a la consola de GRUB ni editar las entradas de arranque a menos que se introduzca la contraseña utilizando la tecla 'P\

Protección de GRUB2

En la segunda versión del gestor de arranque se modifican bastantes aspectos y se apuesta por ofrecer más flexibilidad a la hora de configurarlo. Todo ello mediante Scripts para automatizar configuraciones y nuevas directivas de configuración entre otros cambios. A pesar de ello, el aspecto sigue siendo casi idéntico al de GRUB.

Centrando la atención en la protección mediante ***password***, la principal novedad radica en que es posible la creación de roles o grupos de usuarios con diferentes privilegios en GRUB2. Así pues, es posible disponer del rol por defecto ***superusers*** y agregar a él diferentes usuarios que adquirirán la posibilidad de acceder a la terminal de GRUB y modificar entradas de arranque. Para esta configuración se centrará la atención únicamente en el rol ***superusers.***

Como ocurría en la primera versión de GRUB, puede establecerse el ***password*** en texto plano o cifrado además de bloquear diferentes líneas de arranque. La diferencia es que si se escoge la opción de cifrado, ya no se establecerá en ***md5***, sino que en su defecto se utiliza ***pbkdf2.*** La herramienta a utilizar para generar el ***password*** cifrado es ***grub-mkpasswd-pbkdf2***.

A la hora de establecer el ***password***, en el caso de Debían en concreto, es recomendable hacerlo en el fichero ***/etc/grub.d/40_custom.*** Nuevamente, como recomendación, se ha utilizado un ***password*** cifrado y el fichero final tendrá el siguiente aspecto.

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.

set superusers="admin"
password_pbkdf2 admin grub.pbkdf2.sha512.10000.BC65DD5C796105A485F223BD3FE4113F6
9BA8B7C3FABC101989B39683FC303144482E4109B3F7333548E7D512709F2D2895C0A795B32948DE
5A30B7BCD1BA23B.10A75AFBF5A5E3BFE33BA189173FC17B27D6714AA3397B58F5F18251BA40E4F5
AD40D4010598D80D25798B228C64A0AB76EB1D5AA0C707EAA4FF4955250949BD
```

Fig 2.06: Estableciendo un *password* en GRUB2.

Para que la configuración sea efectiva es necesario regenerar el fichero de configuración de GRUB. Para ello se utiliza el comando ***update-grub***, que leerá todos los Scripts incluyendo el ***/etc/grub.d/40_custom.*** En el momento en que se reinicie la máquina se podrá comprobar que la consola no es

accesible a menos que se introduzcan las credenciales establecidas. De igual modo estará restringido el acceso a la edición de las líneas de arranque. En el caso del ejemplo, habría que introducir el nombre de usuario "admin" y la contraseña establecida.

Si se precisa del bloqueo de arranque de los núcleos ***Linux***, sería necesario modificar el script que se encarga de detectar dichos sistemas y establecer el parámetro ***—users.*** En el caso de Debían se trata del fichero ***/etc/grub.d/1 Olinux***.

3. Protección del sistema de ficheros

Como se ha podido comprobar, la protección del gestor de arranque reduce la zona de impacto cuando un atacante se encuentra físicamente en el mismo emplazamiento de la máquina. También es posible, a pesar de no haberse mencionado previamente, que el atacante haya conseguido acceso de algún modo a un ***KVM***y pueda conectarse a la red para manejar una máquina de forma remota como si se estuviera en local. En caso de ser así el problema se ha zanjado en cuanto a protección física se refiere. El atacante ya no tiene la posibilidad de aprovechar la configuración por defecto de GRUB para obtener acceso al sistema.

Suponiendo la situación en la que el atacante se encuentra físicamente junto a la máquina objeto del ataque, es obvio que surge una pregunta. ¿Y si se apaga la máquina y se extraen los discos físicos instalados en la misma?

Concepto de acceso a un sistema de ficheros

El atacante finalmente ha conseguido extraer los discos de la máquina. Ahora, en algún lugar desconocido donde reside el atacante, monta esos discos en un equipo y arranca su sistema operativo ***Linux*** u otro capaz de abrir diversos tipos de sistemas de ficheros. Para este ejemplo en concreto, el atacante ha conseguido obtener un disco de 20GB de capacidad tipo SATA. A continuación de describen los sencillos pasos para visualizar la información del disco.

Una vez en el sistema anfitrión, que aloja al disco usurpado, se hace uso del comando ***fdisk*** por ejemplo para visualizar la tabla de particiones del disco.

```
Disco /dev/sdb: 17.2 GB, 17179869184 bytes
255 heads, 63 sectors/track, 2088 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0001fd44

Disposit. Inicio    Comienzo       Fin      Bloques   Id  Sistema
/dev/sdb1   *           1        1998    16043008   83  Linux
/dev/sdb2            1998        2089      731137    5  Extendida
/dev/sdb5            1998        2089      731136   82  Linux swap / Solaris
```

Fig 2.07: Tabla de particiones con *fdisk*.

Por el tamaño de las particiones, todo indica que la más suculenta y por lo tanto la que puede albergar más información es la ***/dev/sdbl.*** Se puede estudiar el tipo de partición mediante el comando ***blkid/ dev/sdbl*** en este caso.

Se procede a montar la aplicación para acceder a su contenido. Para este fin se usa el comando ***mount*** como de costumbre.

```
root@debtest2:~# mount /dev/sdb1 /mnt
root@debtest2:~# cat /mnt/etc/passwd | grep root
root:x:0:0:root:/root:/bin/bash
root@debtest2:~# cat /mnt/etc/shadow | grep root
root:$6$MCzuA0iD$ke5OTyCG4/3DWZwrsyPGCk3JI6SVNB45XEjg9/goI2hqFzVF5P38aVK5yyf0B00
NoxxAA8rDlzlODRhmPfYts/:15676:0:99999:7:::
root@debtest2:~# _
```

Fig 2.08: Montando imagen con *mount* y visualizando contenido.

La imagen anterior es solo una muestra de información sensible que puede llegar a conseguirse. Siempre puede ser interesante para un atacante observar en los directorios ***home*** de los usuarios, el directorio ***root***, el directorio ***var***, etc.

Siendo el ***súper-usuario*** de la máquina en la que se ha montado el disco no habrá nada que pueda resistirse a ser visualizado. ¿Qué es lo que podría hacerse para tratar de solventar este problema? ¿Cómo se podría proteger un disco en caso de extravío o robo? La solución a ambos problemas consiste en utilizar cifrado de datos. La mejor solución para ello es utilizar cifrado completo ya sea de discos o de particiones. A continuación se detalla cómo se puede llevar a cabo dicha operación, bastante sencilla de implementar a pesar de lo que pueda parecer.

Cifrado de disco o particiones

Existen diversos métodos para poner en práctica la acción de cifrado de dispositivos o discos en sistemas operativos ***GNU/Linux.*** A continuación se explica ***LUKS***, que se ha convertido en el estándar.

A pesar de ello, otro de los sistemas ampliamente conocidos y perfectamente utilizables es ***TrueCrypt.*** Es un proyecto open-source y entre sus principales ventajas destaca la posibilidad de ser utilizado con sistemas ***Linux***, Windows y MacOS X. Todo ello sin ningún tipo de incompatibilidad. Sin duda, este sistema de cifrado parece más interesante para dispositivos que sean trasladados a menudo y utilizados en sistemas operativos diferentes. Un ejemplo claro podría ser una unidad flash USB o bien un disco duro externo utilizado para almacenar datos.

LUKS. Linux Unified Key Setup

LUKS es una especificación de cifrado que trata de definir un formato estándar de disco, ser independiente de la plataforma utilizada y poseer una extensa y clara documentación.

Además de que LUKS es un estándar como se comentaba con anterioridad, se ha optado por su uso puesto que se incluye en la mayoría de los instaladores de las principales distribuciones, lo que hace

que la acción de cifrar los discos o particiones sea una tarea casi transparente a la hora de instalar el sistema operativo.

Antes de pasar con ejemplos prácticos de implementación, se antoja necesario explicar levemente cómo funciona este sistema. No se trata de profundizar en demasía, pues el mundo de la criptografía es inmenso y muy técnico, sino de ver a grandes rasgos qué componentes entran en juego a la hora de utilizarlo en una distribución ***Linux*** moderna. Dichos componentes son:

- ***Device Mapper, DM.*** Se trata de un framework capaz de mapear dispositivos de bloque dentro de otros. Es decir, se pueden montar los dispositivos que se deseen dentro de otros dispositivos, habiendo siempre un último que se comunique con un dispositivo físico. Si los dispositivos ***DM*** se representasen en forma de pila, en la parte más alta podría encontrarse un sistema de ficheros entre otros.
- ***dm-crypt.*** Se trata de una de las aplicaciones o implementaciones de ***DM {Device Mapper).*** Su cometido consiste en proporcionar cifrado transparente para un dispositivo de bloques y usa como intermediario DM; ya sea entre un disco físico u otro dispositivo de bloques virtual.
- ***LUKS.*** Es el mecanismo utilizado para el cifrado de dispositivos de bloque. En él se especifican algoritmos de cifrado, passphrase, modos de cifrado, vectores de incialización ***(IVs),*** etc.

Para resumir todo lo anterior con un ejemplo, se supone una partición de un disco como puede ser la partición extendida número 5. Se dispone por lo tanto de /dev/sda5 y es la partición que se desea cifrar pues va a contener valiosos documentos que nadie debería poder visualizar en caso de pérdida del disco.

Una vez que se cifre dicha partición, no se volverá a trabaj ar más con /dev/sda5; sino con un dispositivo de bloques virtual proporcionado por dm-crypt en el que se habrá usado LUKS como especificación de cifrado. El nuevo dispositivo de trabajo, en el que habrá que crear un sistema de ficheros, pasaría a localizarse en /dev/mapper/<nombre>. Por supuesto, dicho dispositivo virtual estará disponible una vez que se haya desbloqueado con una frase de paso como se verá posteriormente.

Caso práctico I. Instalar un sistema con cifrado de datos

Como se comentó con anterioridad, la mayoría de las distribuciones disponen de soporte de cifrado de discos en sus instaladores, ya sean gráficos o por texto. En el caso de Debían ***GNU/Linux*** esto no es diferente, y a continuación se mostrará el aspecto que tiene la parte de cifrado en el momento de instalación del sistema operativo. De la misma manera, se verá el resultado obtenido tras la instalación.

Durante la instalación y más en concreto la parte relacionada con el cifrado, se seleccionarán los valores por defecto y más sencillos, para demostrar que es realmente asequible poner en pie un sistema cifrado.

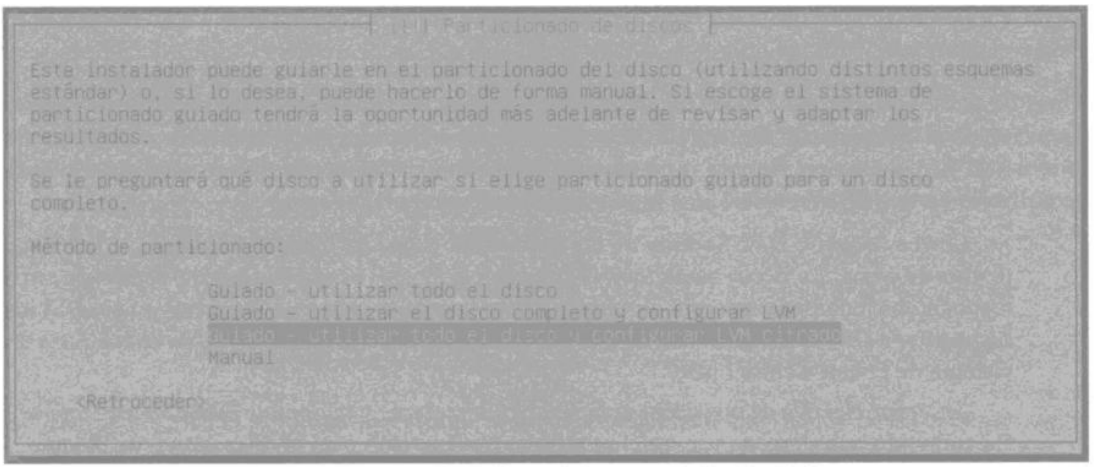


Fig 2.09: Particionado de disco guiado y con cifrado. Muy fácil.

Como se aprecia en la imagen anterior, se ha seleccionado el particionado de discos guiado con cifrado de datos. Posteriormente y para no complicar las cosas se selecciona que el sistema será instalado en una misma partición. Realmente esto último no es lo recomendable para un sistema en producción, sobre todo cuando se generan grandes ficheros de log, se requiere de configuraciones más complejas, etc. Pero de momento, se quiere demostrar cómo quedaría un sistema cifrado.

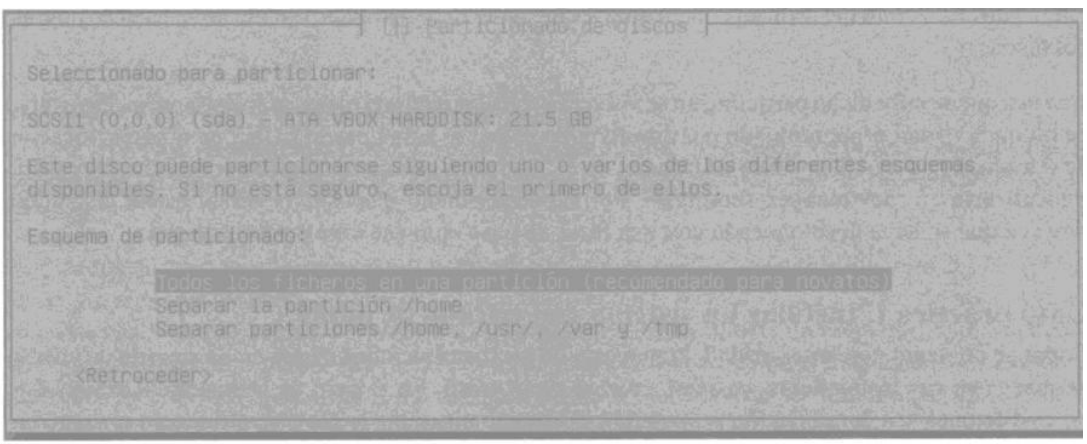


Fig 2.10: Instalación del sistema en una sola partición.

Los sistemas de cifrado pueden configurarse de diferentes modos. Por ejemplo se puede cifrar un partición y posteriormente formatearla como se desee, crear volúmenes lógicos y cifrar los discos lógicos necesarios, hacerlo en un raid hardware o fake-raid, etc.

En el caso por defecto del instalador de Debian se realiza del siguiente modo:

- Partición 1, primaria, sin cifrar. Será la utilizada para montar /boot y que contiene la configuración del gestor de arranque, el núcleo ***Linux***, etc. Debe ser así, pues si no el sistema no sería capaz de arrancar.
- Partición 5, primera partición lógica, cifrada. Esta partición será un grupo de volúmenes lógicos que contendrá dos volúmenes lógicos. Cada volumen a su vez se formatea de forma adecuada. Una partición será la destinada a montar / y la otra será la ***swap*** del sistema. Se entiende que se usan volúmenes lógicos para proporcionar un poco de más flexibilidad al sistema. Para este menester se utiliza ***LVM (Logical Volumen Manager)*** que es otra de las aplicaciones de ***Device Mapper.***

Dependiendo del tamaño del disco y de la velocidad del sistema, es posible que el proceso de borrado de datos del disco ocupe un tiempo considerable. En este proceso se realiza una escritura en el disco de datos aleatorios que "sustituirá" todo lo que había con anterioridad. En el caso del instalador de Debian, la aplicación que realiza dicha acción es ***blockdev-wipe.***

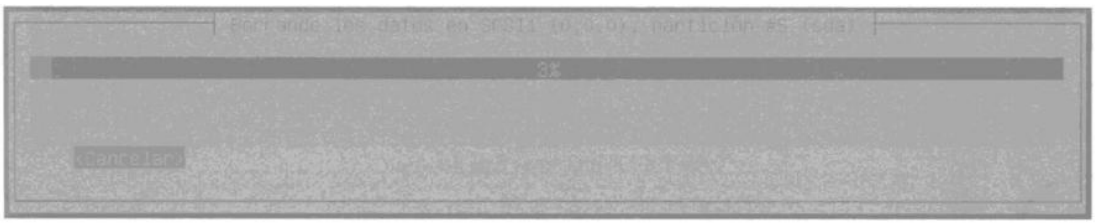


Fig 2.11: Borrado de datos de la partición cifrada.

Una vez que finaliza el proceso anterior se le solicita al usuario la introducción de una frase de paso, passphrase, que servirá para desbloquear la clave maestra del sistema de cifrado proporcionado por ***LUKS.*** Como consecuencia, al introducir dicha frase de paso, la partición quedará desbloqueada y de cara al sistema y al usuario será como una partición convencional.

Al confirmar la frase paso que desbloqueará el sistema en cada arranque, se muestra un sumario de cómo ha quedado distribuido el disco físico.

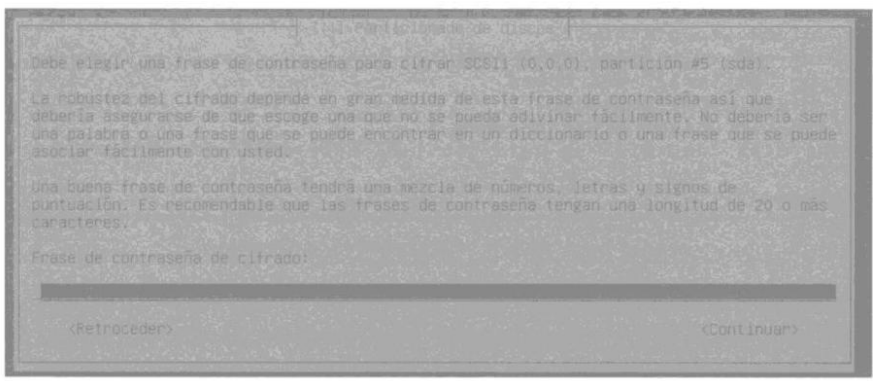


Fig 2.12: Estableciendo la passphrase.

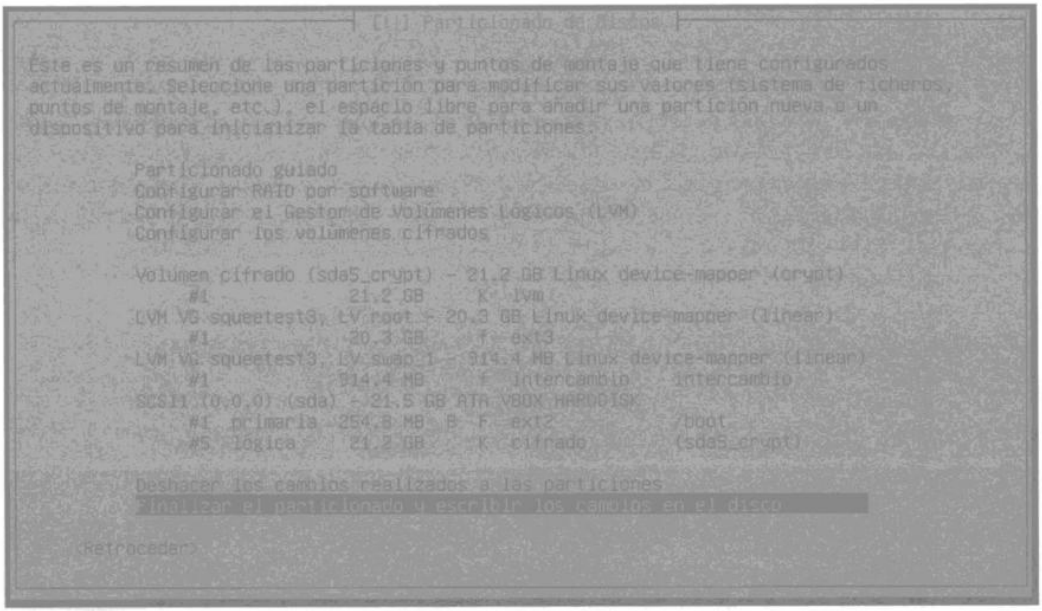


Fig 2.13: Particionado final del disco.

Todo ha quedado tal y cómo se comentó anteriormente. Se ha utilizado sobre una partición cifrada LVM para crear dos volúmenes lógicos que se formatearán como ***ext3*** y ***swap.***

Una vez que finaliza la instalación y la máquina arranca, aparecerá el ya conocido gestor de arranque GRUB2. Al pulsar sobre la entrada del sistema operativo recién instalado con cifrado se iniciará el núcleo ***Linux*** como de costumbre. El iniciar el sistema es necesario montar la partición raíz entre otras, pero en este caso al estar cifrado aparecerá un mensaje como el que sigue:

```
Loading Linux 2.6.32-5-686 ...
Loading initial ramdisk ...
Loading, please wait...
  Volume group "debtest2" not found
  Skipping volume group debtest2
Unable to find LVM volume debtest2/root
  Volume group "debtest2" not found
  Skipping volume group debtest2
Unable to find LVM volume debtest2/swap_1
Unlocking the disk /dev/disk/by-uuid/f66c1c62-ee8e-4bd5-a0fe-0d43a480c5cd (sda5_
crypt)
Enter passphrase: _
```

Fig 2.14: Introducción de la passphrase en el arranque.

Al introducir de forma correcta la frase de paso, la partición cifrada quedará desbloqueada de cara al sistema y se trabajará de forma transparente al cifrado del disco. Tanto ***DM***, ***dm-crypt*** y ***LVM2*** en este caso estarán realizando su magia.

Aplicaciones de gestión y administración para el cifrado con LUKS

En una instalación limpia del sistema operativo sobre un disco cifrado, además de disponer de las herramientas básicas para la gestión de discos, se antojan necesarias herramientas para poder gestionar volúmenes lógicos, volúmenes cifrados, etc.

A continuación, haciendo uso de las herramientas se pasa a detectar si efectivamente el sistema corre sobre una partición cifrada.

```
root@squeetest3:~# blkid
/dev/sda5: UUID="a4f0b0b8-ed81-4aa0-bff1-13f2ad88e5bb" TYPE="crypto_LUKS"
/dev/sda1: UUID="cadaf0ff-4ef0-40b5-9775-a8f17d74a973" TYPE="ext2"
/dev/mapper/sda5_crypt: UUID="SG22s3-xIcM-D8SU-zpLn-MS91-tcA9-bR60SK" TYPE="LVM2
_member"
/dev/mapper/squeetest3-root: UUID="0fba3962-fe1e-4bfd-942b-8fc4ed3f79f0" TYPE="e
xt3"
/dev/mapper/squeetest3-swap_1: UUID="8255ae88-7539-4e42-b8f6-9510504fa6ad" TYPE=
"swap"
root@squeetest3:~# _
```

Fig 2.15: Mostrando tipos de particiones con *blkid.*

```
/dev/mapper/squeetest3-root on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sda1 on /boot type ext2 (rw)
```

Fig 2.16: Puntos de montaje del sistema.

El comando ***blkid*** ayuda a identificar el tipo de contenido de las particiones detectadas por el sistema. En el caso anterior puede observarse como ha detectado que la partición 5 del disco, ***/dev/sda5***, está cifrada con ***LUKS.*** Se puede continuar atendiendo a los montajes de directorios del sistema y particiones. Por ejemplo, dónde está montada la raíz (/). Para ello se utiliza el comando ***mount.***

Se observa en la imagen anterior cómo las particiones raíz y swap están montadas en dos volúmenes lógicos. Como se comentó, la configuración se establece de ese modo por defecto en la instalación de Debían cuando se elige cifrado de datos. Tanto los volúmenes lógicos, como los volúmenes cifrados están funcionando gracias a ***Device Mapper.*** Se puede visualizar el estado de los llamados dispositivos ***dm*** con la siguiente serie de comandos.

```
root@squeetest3:~# dmsetup info | awk '/^Name/||/^Major/{print $0}'
Name:              squeetest3-root
Major, minor:      254, 1
Name:              squeetest3-swap_1
Major, minor:      254, 2
Name:              sda5_crypt
Major, minor:      254, 0
root@squeetest3:~# ls -al /dev/dm*
brw-rw---- 1 root disk 254, 0 dic  7 22:01 /dev/dm-0
brw-rw---- 1 root disk 254, 1 dic  7 22:01 /dev/dm-1
brw-rw---- 1 root disk 254, 2 dic  7 22:01 /dev/dm-2
root@squeetest3:~# ls -l /dev/mapper/*
crw------- 1 root root 10, 59 dic  7 22:01 /dev/mapper/control
lrwxrwxrwx 1 root root      7 dic  7 22:01 /dev/mapper/sda5_crypt -> ../dm-0
lrwxrwxrwx 1 root root      7 dic  7 22:01 /dev/mapper/squeetest3-root -> ../dm-
1
lrwxrwxrwx 1 root root      7 dic  7 22:01 /dev/mapper/squeetest3-swap_1 -> ../d
m-2
```

Fig 2.17: Volúmenes generados con DM, *Device Mapper.*

El volumen ***/dev/mapper/sda5_crypt o /dev/dm-0*** es el correspondiente al volumen cifrado que se creó durante la instalación del sistema. Los otros dos volúmenes son la partición raíz y la ***swap***, que se tratan en realidad de dos volúmenes lógicos contenidos en un grupo de volúmenes usando el "disco físico" ***/dev/dm-0*** que se trata en realidad del volumen cifrado. Aquí es donde entra enjuego el concepto en el que ***Device Mapper*** es capaz de mapear dispositivos de bloque dentro de otros. A pesar de que no es objetivo explicar aquí la gestión de los volúmenes lógicos con ***Ivm***, a continuación se utilizan los comandos ***pvs***, ***vgs*** y ***Ivs*** para mostrar toda la información relevante al respecto.

```
root@squeetest3:~# pvs
  PV         VG         Fmt  Attr PSize  PFree
  /dev/dm-0  squeetest3 lvm2 a-   19,76g    0
root@squeetest3:~# vgs
  VG         #PV #LV #SN Attr   VSize  VFree
  squeetest3   1   2   0 wz--n- 19,76g    0
root@squeetest3:~# lvs
  LV     VG         Attr   LSize   Origin Snap%  Move Log Copy%  Convert
  root   squeetest3 -wi-ao  18,91g
  swap_1 squeetest3 -wi-ao 872,00m
```

Fig 2.18: Estado de los volúmenes lógicos.

De acuerdo, ¿pero qué aspecto tiene el fichero ***/etc/fstab*** ahora? - La respuesta es sencilla. Se parece mucho al de un sistema sin cifrar, salvo que en este caso se usarán los dispositivos de bloque del contenido en ***/dev/mapper*** para las particiones raíz y ***swap.***

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
proc            /proc           proc    defaults        0       0
/dev/mapper/squeetest3-root /                   ext3    errors=remount-ro 0       1
# /boot was on /dev/sda1 during installation
UUID=cadaf0ff-4ef0-40b5-9775-a8f17d74a973 /boot           ext2    defaults        0       2
/dev/mapper/squeetest3-swap_1 none            swap    sw              0       0
/dev/scd0       /media/cdrom0   udf,iso9660 user,noauto     0       0
```

Fig 2.19: Fichero */etc/fstab*.

Muy bien, pero el sistema raíz se encuentra cifrado con ***LUKS,*** por lo que es necesario indicar de algún modo que es necesario desbloquear la partición para empezar a trabajar con normalidad. Es decir, es necesario crear los volúmenes lógicos y montar las particiones. El fichero en el que se indican qué particiones cifradas con ***dm-crypt*** deben desbloquearse es el ***/etc/crypttab.*** En dicho fichero se definen los dispositivos de bloque cifrados y bajo qué nombre deben aparecer en el directorio /***dev/mapper***.

Aun así, sigue habiendo algo que no cuadra en el caso de una partición raíz con el sistema completo cifrado. ¿Cómo es posible saber qué dispositivo cifrado hay que desbloquear si ***/etc/crypttab*** está cifrado? La respuesta está en que al instalar el sistema con cifrado se ha generado la ***ramdisk*** o ***initramfs*** configurada de tal forma que sea capaz de iniciar el proceso de desbloqueo de la partición ***/dev/sda5.*** De ese modo, cuando el núcleo ***Linux*** carga la ***ramdisk*** se solicita la ***passphrase*** de

desbloqueo. Una conclusión que se puede sacar de todo esto es que la línea correspondiente a ***/dev/sda5***, en este ejemplo, del fichero ***/etc/crypttab*** es innecesaria.

Caso práctico II. Crear una partición cifrada para datos en un sistema existente

Quizás la aplicación principal y más importante para el cifrado con ***dm-crypt*** es ***cryptsetup.*** Si el sistema no dispone de dicha herramienta puede instalarse desde los repositorios oficiales descargando el paquete ***cryptsetup.***

Se supone un sistema instalado y totalmente funcional en el que se ha instalado un disco de 11GB con una partición de 8GB destinada al cifrado. Por encima de la capa de cifrado se creará un sistema de ficheros de tipo ***ext4*** destinado al almacenamiento de datos personales. Por otra parte se verá cómo gestionar varias contraseñas de desbloqueo del dispositivo cifrado.

```
root@squeetest3:~# fdisk -l /dev/sdb

Disco /dev/sdb: 10.7 GB, 10737418240 bytes
118 heads, 33 sectors/track, 5385 cylinders
Units = cilindros of 3894 * 512 = 1993728 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x2e954724

Disposit. Inicio     Comienzo       Fin       Bloques   Id  Sistema
/dev/sdb1                1        4310     8388608    83  Linux
```

Fig 2.20: Partición del disco secundario para el cifrado.

A continuación se procede a preparar la partición para su uso como contenedor cifrado mediante ***dm-crypt*** + ***LUKS.*** Es recomendable llenar la partición o dispositivo a cifrar para tratar de dificultar posibles ataques que puedan llevarse a cabo. En este caso se pasa a crear directamente el contenedor ***LUKS.***

```
root@squeetest3:~# cryptsetup luksFormat /dev/sdb1

WARNING!
========
This will overwrite data on /dev/sdb1 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
```

Fig 2.21: Estableciendo la partición */dev/sdb1* como contenedor LUKS.

Como se puede observar se trata de un proceso muy sencillo y con el que se genera el contenedor ***LUKS,*** junto con su información de cifrado por defecto, la clave maestra, así como una ***passphrase*** inicial con la que se cifra una copia de la clave maestra. Para comprobar que efectivamente la partición ***/dev/sdbl*** se trata ahora de un contenedor LUKS se puede utilizar nuevamente el comando ***cryptsetup*** con un argumento diferente.

```
der information for /dev/sdb1

        1
ame:    aes
ode:    cbc-essiv:sha256
:       sha1
offset: 2056
        256
:       ed c3 dd ba 6e bd 2b d0 61 4b 4e 01 3b 9f 8f 5a 62 e2 73 fe
        3a 58 b6 01 cd 4a bd 53 76 05 4c 31 b0 73 4f 0c
        49 0b 0c fc 6c 51 0d 6c 0d 80 9e 2d ea 06 ef ab
tions:  54250
        969276bf-7ca2-45e5-8a53-d9b5adba28a8

 0: ENABLED
Iterations:             217314
Salt:                   47 58 b8 a5 9f 52 85 87 ad 63 c4 00 ec 24 c6 d2
                        ee 9c ef ac a4 d5 91 73 03 be 37 a8 77 e1 a3 16
Key material offset:    8
AF stripes:             4000
 1: DISABLED
 2: DISABLED
 3: DISABLED
 4: DISABLED
```

Fig 2.22: Información de *LUKS*.

La salida anterior muestra información sobre el tipo de cifrado, el ***checksum*** de la clave maestra (***MK)*** y los ***key-slots.*** En efecto, cada ***key-slot*** conforma la zona en la que se almacena el hash de la passphrase escogida. En ***LUKS*** existen ocho ***key-slots***, del cero al siete tal y como se comprueba en la imagen. Esta funcionalidad dota de aún más flexibilidad a este sistema de cifrado, pues diferentes personas que requieran del desbloqueo de un disco podrán hacerlo con su propia passphrase.

Los parámetros existentes en cryptsetup para gestionar las passphrases son ***luksAddKey***, ***luksDelKey*** y ***luksKillSlot.*** En caso de la necesidad de agregar otra passphrase se puede realizar del siguiente modo. Nótese la salida de la cabecera ***LUKS*** del disco ya que habrá otro ***key-slot*** utilizado.

```
root@squeetest3:~# cryptsetup luksAddKey /dev/sdb1
Enter any passphrase:
Enter new passphrase for key slot:
Verify passphrase:
root@squeetest3:~# cryptsetup luksDump /dev/sdb1
LUKS header information for /dev/sdb1

Version:        1
Cipher name:    aes
Cipher mode:    cbc-essiv:sha256
Hash spec:      sha1
Payload offset: 2056
MK bits:        256
MK digest:      ed c3 dd ba 6e bd 2b d0 61 4b 4e 01 3b 9f 8f 5a 62 e2 73 fe
MK salt:        3a 58 b6 01 cd 4a bd 53 76 05 4c 31 b0 73 4f 0c
                49 0b 0c fc 6c 51 0d 6c 0d 80 9e 2d ea 06 ef ab
MK iterations:  54250
UUID:           969276bf-7ca2-45e5-8a53-d9b5adba28a8

Key Slot 0: ENABLED
```

Fig 2.23: Creando una segunda passphrase.

```
Key Slot 0: ENABLED
        Iterations:             217314
        Salt:                   47 58 b8 a5 9f 52 85 87 ad 63 c4 00 ec 24 c6 d2
                                ee 9c ef ac a4 d5 91 73 03 be 37 a8 77 e1 a3 16
        Key material offset:    8
        AF stripes:             4000
Key Slot 1: ENABLED
        Iterations:             218007
        Salt:                   5d 30 00 33 e3 e0 c8 59 f9 bb e9 93 23 a6 95 ac
                                d9 77 4f 92 a3 22 8a 7d 02 d7 44 2d cd b3 66 dc
        Key material offset:    264
        AF stripes:             4000
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

Fig 2.23: Creando una segunda passphrase (continuación).

Apartir de este momento el disco podrá ser desbloqueado por dos passphrases diferentes. Obviamente, para establecer una nueva passphrase es necesario introducir una establecida previamente que pueda desbloquear la clave maestra tal y como se aprecia en la imagen.

Para poder usar la nueva partición ***LUKS*** es necesario desbloquearla, formatearla y configurar el sistema para que en cada arranque solicite su desbloqueo aparte de montarla de forma automática. En este caso sí resulta necesario el fichero ***/etc/crypttab*** al que habrá que agregarle una entrada como la siguiente:

```
sdb1enc UUID=a4f0b0b8-ed81-4aa0-bff1-13f2ad88e5bb none luks
```

El ***UUID*** del disco puede ser fácilmente extraído realizando un listado del directorio ***/dev/disk/ by-uuid*** por ejemplo. El primer parámetro corresponde al nombre con el que se mapeará la unidad ***LUKS*** desbloqueada. Con dicha línea, se solicitará la introducción de una de las passphrases en el próximo arranque. Si el desbloqueo resulta exitoso el dispositivo de bloques estará disponible en / ***dev/mapper/sdblenc*** en este ejemplo.

Aún resta formatear el dispositivo de bloques virtual y establecer su montaje automático en el arranque. Para ello, en lugar de reiniciar la máquina se puede utilizar el siguiente comando:

```
cryptsetup luksOpen /dev/sdb1 sdb1enc
```

El dispositivo ***/dev/mapper/sdblenc*** pasa a estar disponible en este momento a falta tan solo de proporcionarle formato. Puede realizarse del siguiente modo.

```
mkfs.ext4 /dev/mapper/sdb1enc
```

Lo siguiente, antes de reiniciar, consiste en agregar la línea en el fichero ***/etc/fstab*** encargada de montar la partición en el directorio ***/mnt*** por ejemplo.

```
/dev/mapper/sdb1enc   /mnt    ext4    errors=remount-ro     0       2
```

En el próximo reinicio la máquina se quedará congelada en un mensaje como el que sigue. En él se solicita una de las passphrases para desbloquear la unidad.

```
Starting early crypto disks...sdb1enc (starting)...
Unlocking the disk /dev/disk/by-uuid/969276bf-7ca2-45e5-8a53-d9b5adba28a8 (sdb1e
nc)
Enter passphrase: _
```

Fig 2.24: Desbloqueo de la unidad de datos.

```
root@squeetest:~# blkid
/dev/sda1: UUID="9d7e67ca-c0d8-4060-8a85-871dd382ff5c" TYPE="ext3"
/dev/sda5: UUID="f2163b48-0b6e-44a4-8bcf-30cc640fcc19" TYPE="swap"
/dev/sdb1: UUID="969276bf-7ca2-45e5-8a53-d9b5adba28a8" TYPE="crypto_LUKS"
root@squeetest:~# mount /dev/sdb1 /mnt
mount: tipo de sistema de ficheros 'crypto_LUKS' desconocido
```

Fig 2.25: Montar partición cifrada de forma tradicional.

Situación conseguida tras el cifrado de discos

A partir de este momento si los discos físicos del sistema fueran sustraídos, no sería tan trivial conseguir acceso al sistema de ficheros. ¿Qué ocurre si se trata de montar una partición cifrada con LUKS de forma tradicional?

El comando ***mount*** ha finalizado mostrando un error. Como se suponía, no es posible montar una partición cifrada como si se tratase de una partición convencional. Atendiendo a lo comentado anteriormente, primero habrá que desbloquear la unidad para posteriormente trabajar sobre el dispositivo de bloques virtual generado.

4. Cifrado de ficheros

Hasta el momento se ha conseguido cifrar el sistema de ficheros o mejor dicho las particiones que lo contienen. El método resulta efectivo, pero aún puede existir alguna posibilidad más de conseguir acceso a información privilegiada estando físicamente en el emplazamiento del servidor.

En numerosas ocasiones, tanto usuarios como administradores de sistemas olvidan cerrar sus sesiones o simplemente no las bloquean. Si se diera este caso, todas las prácticas seguidas hasta el momento serían en vano. Por muy fuerte que sea el cifrado de las particiones y su passphrase, se haya protegido el gestor de arranque, etc. no servirá de nada para esta situación pues el sistema está completamente desbloqueado y funcional.

Existen diversos métodos para tratar de mitigar este problema, pero el que aquí se presenta consiste en cifrar ficheros sensibles mediante el uso de ***GPG***, ***GNU Privacy Guard.***

Sobre GPG y su modo de funcionamiento

GPG, ***GNUPrivacy Guará***, es una implementación del estándar ***OpenPGP***, que a su vez nació como versión libre de ***PGP***, ***Pretty GoodPrivacy.*** Se encuentra disponible por defecto en la mayoría de las distribuciones basadas en ***Linux*** hoy en día y en caso contrario, está disponible en los repositorios oficiales de cada distribución.

Mediante el uso de la aplicación ***gpg*** se pueden realizar las siguientes acciones:

- Firmado de ficheros mediante el uso de clave privada. Su finalidad consiste en determinar si el fichero firmado pertenece a quien dice ser.
- Cifrado de ficheros mediante el uso de clave pública. Consiste en proteger un fichero cifrándolo por completo haciendo uso de una clave pública.
- Cifrado de ficheros mediante el uso de una passphrase. Es exactamente igual que el caso anterior salvo que se usa cifrado simétrico. Una misma clave para cifrar y descifrar.

La última acción es la más conocida y sencilla de llevar a cabo, pero como contrapartida no resulta tan flexible como su homónima de clave pública.

Antes de continuar con la utilización de ***gpg***, resulta necesario hacer una pequeña introducción al funcionamiento de la criptografía de clave pública.

Funcionamiento básico de la criptografía de clave pública

Consiste en un método de por el cual se utilizan un par de claves dependientes la una de la otra. En las dos claves están la privada y la pública. Como su nombre indica, la pública puede ser accesible por cualquier persona y la privada solo debería estar en poder de la persona a la que pertenezca.

Aunque no es obligatorio, la clave privada debe disponer de una passphrase para desbloquear su uso. Si cayera en manos no deseadas y no estuviera protegida se podría llevar a cabo una suplantación de identidad y acceso a contenido sensible.

Los usos de estas claves, tal y como se comentaba anteriormente son el de cifrado y firmado de ficheros.

Cifrado y descifrado de ficheros

La finalidad de cifrar es la de mantener la privacidad de los datos. Puede ser para compartir el mensaje o archivo con una persona de confianza o bien para protegerlos en un disco y desbloquearlos cuando sea necesario. El proceso para el cifrado sería el siguiente:

1. Se cifra el contenido con la clave pública, por lo que cualquier persona con acceso a la clave pública puede cifrar el contenido de un mensaje o fichero.
2. El proceso de descifrado se realiza mediante la pareja de la clave pública que ha cifrado el mensaje o archivo. Es decir, es necesario utilizar la clave privada asociada a la clave pública

que ha cifrado. Si la clave privada se generó con una passphrase asociada será necesaria su introducción para acceder al mensaje o archivo.

Firmado y verificación de ficheros

El firmado de un contenido es una acción que puede realizarse tanto si el fichero o mensaje ha sido cifrado tanto como si no. Se utiliza para poder verificar que la fuente de la cual proviene un archivo o mensaje es realmente la que dice ser. El método para firmar es el siguiente:

1. Se firma el archivo o mensaje con la clave privada. Debe introducirse la passphrase que bloquea la clave en caso de disponer de ella.
2. Cualquier persona con acceso a la clave pública puede verificar la firma y comprobar si el origen del mensaje o archivo es correcto.

La base para el cifrado mediante criptografía de clave pública reside en que para cada una de las claves pública y privada solo puede existir su pareja privada y pública respectivamente. Es por ello que si se en todo momento la clave privada está a salvo o bien protegida mediante passphrase no existe peligro alguno.

En el caso de GPG no existe ninguna autoridad certificadora, ***CA*** como ocurre con las implementaciones ***PK1***, ***Prívate Key Infraestructure.*** Para ***GPG*** y se usan las llamadas relaciones de confianza, que consiste en que los propios usuario verifican que una clave pública accesible por todos pertenece a la persona indicada por la clave.

Una vez que se ha explicado el funcionamiento de ***GPG*** a grosso modo, se exponen dos casos prácticos para proteger la información sensible de un disco.

Cifrado simétrico con GPG

Para este ejemplo se pretende cifrar un fichero de texto plano que contiene información muy importante. Para ello se hará uso del cifrado simétrico con ***gpg*** usando el parámetro ***—symmetric.***

```
root@squeetest:~# gpg --symmetric top_secret.txt
Introduzca frase contraseña: _
```

Fig 2.26: Cifrado del fichero *top_secret.txt.*

El fichero se ha cifrado y se ha generado su versión cifrada con extensión **<*.gpg*>.** Obviamente, debe eliminarse el fichero no cifrado para disponer de la copia cifrada únicamente. Si se trata de realizar un volcado del fichero se observa que la salida es "basura" o información aleatoria.

```
root@squeetest:~# cat top_secret.txt.gpg
♦♦♦♦♦♦=♦`♦y♦3@m♦♦♦v♦?u♦RC2W♦[9iLC♦♦♦=JKE♦♦♦♦}M♦+8♦♦♦♦ATm♦♦♦Ut♦~♦E♦♦♦c♦L\_U♦♦♦'♦♦
4♦♦♦J+♦♦#♦♦t♦D♦3♦♦N♦♦ ♦root@squeetest:~#
root@squeetest:~# file top_secret.txt.gpg
top_secret.txt.gpg: DOS executable (COM)
```

Fig 2.27: Volcado del fichero *top_secret.txt.gpg.*

No es posible identificar el tipo de fichero ni visualizar la información que contiene. Para descifrar el fichero se vuelve a usar el comando ***gpg*** pasando los parámetros ***—decrypt*** y ***-o.*** Respectivamente para descifrar el fichero y volcarlo en el nombre que se indique.

```
root@squeetest:~# gpg --decrypt -o top_secret.txt top_secret.txt.gpg
gpg: datos cifrados CAST5
gpg: cifrado con 1 frase contraseña
gpg: ATENCIÓN: la intgridad del mensaje no está protegida
root@squeetest:~# cat top_secret.txt
 ------------------------
< Las vacas hablamos... >
 ------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Fig 2.28: Descifrando el fichero cifrado *top_secret.txt.gpg.*

En la imagen anterior puede percibirse que la propia aplicación ***gpg*** avisa de que el contenido puede ser no íntegro. Esto es porque el mensaje se cifró con clave simétrica pero no se firmó su contenido. El motivo, además de simplificar el proceso, es porque aún no se han generado el par de claves pública y privada.

Cifrado asimétrico con GPG

A continuación se realiza el mismo proceso que en el caso anterior salvo que esta vez se usará la implementación de cifrado asimétrico o criptografía de clave pública de ***gpg.***

```
root@squeetest:~# gpg --gen-key
gpg (GnuPG) 1.4.10; Copyright (C) 2008 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Por favor seleccione tipo de clave deseado:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sólo firmar)
   (4) RSA (sólo firmar)
Su elección: 1
las claves RSA pueden tener entre 1024 y 4096 bits de longitud.
¿De qué tamaño quiere la clave? (2048)
El tamaño requerido es de 2048 bits
Por favor, especifique el periodo de validez de la clave.
         0 = la clave nunca caduca
      <n>  = la clave caduca en n días
      <n>w = la clave caduca en n semanas
      <n>m = la clave caduca en n meses
      <n>y = la clave caduca en n años
¿Validez de la clave (0)? 1y
La clave caduca dom 08 dic 2013 16:22:08 CET
¿Es correcto? (s/n) s
```

Fig 2.29: Generación de claves privadas y públicas.

```
¿Es correcto? (s/n) s

Necesita un identificador de usuario para identificar su clave. El programa
construye el identificador a partir del Nombre Real, Comentario y Dirección
de Correo Electrónico de esta forma:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Nombre y apellidos: Carlos Álvarez Martín
Dirección de correo electrónico: calvarez@informatica64.com
Comentario:
Está usando el juego de caracteres `utf-8'.
Ha seleccionado este ID de usuario:
    "Carlos Álvarez Martín <calvarez@informatica64.com>"

¿Cambia (N)ombre, (C)omentario, (D)irección o (V)ale/(S)alir? V
Necesita una frase contraseña para proteger su clave secreta.

Repita frase contraseña:
```

Fig 2.29: Generación de claves privadas y públicas (continuación).

Dependiendo de la actividad del sistema, se necesitará más o menos ***entropía*** y el proceso de generación de la clave será más o menos duradero. Es necesario producir actividad en el sistema para que la generación aleatoria de la clave sea lo suficientemente aleatoria.

En este caso de ejemplo se han utilizado los valores por defecto salvo la caducidad de la clave. Es decir, se ha usado RSA como algoritmo tanto para el cifrado como para la firma, con una longitud de 2048 bits y una caducidad de un año. Cuando el proceso finaliza se generan los keyrings en la carpeta personal del usuario. A partir de este momento se podrán importar, exportar, revocar, etc. las claves públicas o privadas.
En este caso, continuando con el ejemplo, se cifraría el archivo anterior con la orden:

```
gpg --encrypt -r calvarez@informatica64.com top_secret.txt
```

No se ha solicitado ninguna contraseña o frase de paso ya que el cifrado se realiza mediante la clave pública. En el proceso de descifrado, al ser realizado con la clave privada, sí será necesario introducir la frase de paso. Como ocurría en el caso del cifrado asimétrico, el resultado obtenido ha sido el mismo y no será posible visualizar el contenido del fichero a menos que se descifre. Para el descifrado se usará nuevamente la misma orden que en el caso de clave simétrica.

```
root@squeetest:~# gpg --decrypt -o top_secret.txt top_secret.txt.gpg

Necesita una frase contraseña para desbloquear la clave secreta
del usuario: "Carlos Álvarez Martín <calvarez@informatica64.com>"
clave RSA de 2048 bits, ID 43D070A2, creada el 2012-12-08 (ID de clave primaria 5
A5B679E)

gpg: cifrado con clave RSA de 2048 bits, ID 43D070A2, creada el 2012-12-08
      "Carlos Álvarez Martín <calvarez@informatica64.com>"
```

Fig 2.30: Descifrando fichero con clave privada.

Después de ello, el fichero de texto plano del ejemplo volverá a estar disponible para su visualización o edición.

Puede parecer que el cifrado individual de ficheros resulta una tarea pesada, pero existen Scripts, aplicaciones, etc. que hacen uso de ***gpg*** y el hecho de tener que utilizar ficheros cifrados se volverá una tarea más transparente.

5. Otras protecciones

Existen algunas opciones adicionales con las que es posible proteger aún más una máquina frente a ataques físicos. A continuación se enumeran algunas de ellas.

Bloqueo y cierre automático de sesión

Como se comentó con anterioridad, era necesario en numerosas ocasiones cifrar los ficheros relevantes o con información sensible de forma individual. En numerosas ocasiones las sesiones permanecen desbloqueadas y cualquier persona con acceso físico al servidor podrá explotar los privilegios que la sesión le conceda; en la mayoría de casos, acceso como super-usuario.

Variable de entorno TMOUT

Por defecto, la mayoría de distribuciones utilizan ***BASH*** como intérprete de comandos. Dicho intérprete dispone de una variable llamada ***TMOUT*** que está deshabilitada inicialmente. Su cometido consiste en cerrar una sesión inactiva en el tiempo que se le haya asignado como valor expresado en segundos. Un ejemplo de bloqueo de la sesión en 30 segundos de inactividad podría ser el siguiente:

```
root@squeetest:~# export TMOUT=30
root@squeetest:~# ha expirado mientras esperaba alguna entrada: auto-logout

Debian GNU/Linux 6.0 squeetest tty1

squeetest login: _
```

Fig 2.31: Auto log-out con TMOUT.

Para que el funcionamiento anterior sea permanente tras cada reinicio de la máquina, es necesario establecer la variable en cada inicio de sesión. Para ello debe modificarse el fichero ***-/.bashrc*** estableciendo la línea de la imagen anterior.

Aplicación vlock

Si por el contrario se desea poder bloquear la sesión sin la necesidad de cerrarla, puede optarse por la utilización de la aplicación ***vlock.*** No está instalada por defecto en el sistema pero puede obtenerse e instalarse fácilmente a través de los repositorios oficiales. Dispone de varios parámetros para modificar su comportamiento. Como cualquier aplicación, dispone de ayuda para entender cada modo. Al ser extremadamente sencillo de utilizar, no es necesario profundizar en su uso.

En el caso de querer bloquear todas las ***TTY,*** se puede ejecutar el comando ***vlock -a*** que hará que en las terminales aparezca un mensaje como el siguiente.

Fig 2.32: Consolas bloqueadas con *vlock*.

Evitando ei reinicio y apagado accidental

En la mayoría de las ocasiones, las distribuciones están configuradas para responder a las señales de determinadas teclas o botones. En este caso lo que se trata de realizar consiste en inutilizar el botón de apagado y la secuencia de teclas ***Ctrl+Alt+Del.***

Para deshabilitar el apagado se debe ejecutar el siguiente comando.

```
chmod a-x /etc/acpi/powerbtn-acpi-support.sh
```

Se han eliminado los permisos de ejecución para todos los usuarios, lo que imposibilitará que el script se ejecute y como consecuencia la máquina no responderá al botón de apagado. Para evitar el reinicio mediante la combinación de teclas ***Ctrl+Alt+Del*** debe editarse el fichero ***/etc/inittab*** y comentar la siguiente línea.

```
ca:12345:ctrlaltdel:/sbin/shutdown -t1 -a -r now
```

En el momento que se reinicie la máquina o bien se ejecute el comando ***init q***, la máquina no responderá a la pulsación de la combinación teclas mencionada.

Deshabilitar dispositivos de almacenamiento USB

En un servidor no suele ser habitual, salvo casos excepcionales, tener que pinchar un soporte USB para usarlo como almacenamiento externo. Si dicha funcionalidad no es usada, será mejor deshabilitarla para tratar de reducir aún más la posible superficie de impacto.

Existe un método muy sencillo para realizar esta operación, aunque podría hacerse de un modo más elegante utilizando reglas ***udev*** por ejemplo. En este caso se opta por el método más sencillo y obvio. Lo que se debe hacer es poner la lista negra de módulos el encargado de inicializar los dispositivos USB de almacenamiento. El módulo en cuestión para esta ocasión es ***usbstorage.*** Agregando la línea ***blacklist usb storage*** al final del fichero ***/etc/modprobe.d/blacklist*** se conseguirá que el módulo no sea cargado automáticamente y como consecuencia el dispositivo USB de almacenamiento sea reconocido. La línea puede agregarse del siguiente modo:

```
echo "blacklist usb_storage" >> /etc/modprobe.d/blacklist
```

Capítulo III
Protección perimetral

Tras haber estudiado ciertos métodos para proteger la capa física de un entorno, llega el momento de implementar protecciones en la capa más externa en cuanto a la red de comunicaciones se refiere. En este capítulo se verán herramientas que ayudarán a establecer barreras para aumentar la seguridad de la red, habilitando al mismo tiempo accesos remotos seguros. Por otra parte, se expondrá cómo se puede monitorizar el estado de salud de las máquinas del entorno de red con lo que siempre se podrá obtener información de las máquinas, si responden, sus servicios están activos, etc.

l.iptables

Existen muchos métodos de implementar cortafuegos o filtros en ***Linux***, desde el nivel de enlace con ***ebtables*** y ***arptables***, hasta control en el nivel de aplicaciones como podría ser ***Squid*** para cachear y filtrar contenido web. En esta ocasión se tratará el filtrado y manipulación del tráfico a nivel de red y transporte mediante ***iptables.***

¿Qué es iptables?

iptables es una aplicación de nivel de usuario que permite la gestión, configuración y manejo del filtrado de tráfico de red en una máquina ***Linux.*** Para ello se utiliza ***netfilter***, que no es más que un framework incluido en el núcleo ***Linux*** capaz de manipular paquetes de datos de red en diferentes estados. Como información adicional cabe destacar que ***netfilter*** también se refiere al proyecto que ofrece herramientas libres para gestionar cortafuegos bajo ***Linux.***

Funcionamiento de iptables

El kemel ***Linux*** posee la capacidad de hacer pasar los paquetes de red por una serie de reglas e ***iptables*** permite la configuración de ellas. Dichas reglas se agrupan en cadenas y a su vez las cadenas están contenidas por tablas. Las reglas son un conjunto de parámetros en los que se trata de hacer que un datagrama de red coincida atendiendo al protocolo, IPs destino y origen, estado, etc.

Inicialmente al entrar paquetes en el sistema configurado con ***iptables*** pasan por todas las tablas y cadenas configuradas por defecto.

Cuando un datagrama coincide con la condición de una regla, se pasa a tomar una decisión de qué hacer con dicho paquete. Las más comunes son por ejemplo dejar pasar o bien desechar el paquete. Dichas acciones, que pasarán a ser explicadas más adelante, son conocidas como ***ACCEPT***y ***DROP*** respectivamente.

Si no existe ninguna regla que coincida con un paquete de red, a este se le aplicará la política por defecto que tenga la cadena por la que está pasando. Por defecto, como se comentaba con anterioridad, los paquetes pasan por todas las tablas ya que la política por defecto de las cadenas cuando ***iptables*** no está configurado es ***ACCEPT.*** En tal caso, se entiende que se dejan pasar los paquetes que no coincidan con ninguna regla.

Decisión de enrutamiento

Se ha descrito el funcionamiento de ***iptables.*** Pero, ¿qué toma la decisión de por dónde deben pasar, salir o entrar los paquetes de datos?

Ahí es donde entran en juego las tablas de enrutamiento de la máquina. Se atienden a las redes con las que se está directamente conectado, puertas de enlace por defecto, interfaces, etc.

Dependiendo de las cadenas y las tablas, descritas a continuación, las diferentes reglas contenidas en ellas se aplicarán antes o después de la decisión de enrutamiento.

Todo esto será un poco más gráfico con el diagrama de la ***figura*** 7, mostrado al final de la siguiente sección que explica las tablas por defecto y sus respectivas cadenas.

Tablas

Existen cuatro tablas por defecto. Es posible cambiar esa situación mediante la carga de módulos en ***iptables***, por lo que se pueden agregar más tablas.

Las tablas por defecto son:

- **filter.** Es la tabla encargada del filtrado de paquetes. Es decir, en esta tabla se pueden dejar pasar o descartar paquetes dependiendo de las reglas contenidas en sus cadenas predefinidas por defecto enumeradas a continuación:
  - INPUT. Es la cadena en la que se definen las reglas para los paquetes que recibe un proceso local de la máquina. Es decir, los paquetes que tengan como destino la máquina local en la que se configura ***iptables.***
  - OUTPUT. El mismo caso que ***INPUT*** pero a la inversa. Por esta cadena pasarán los paquetes generados por un proceso local de la máquina.
  - FORWARD. Esta cadena es por la que pasan los paquetes que no van dirigidos a la máquina local. Es decir, esta tabla se usa cuando la máquina configurada con ***iptables*** posee la capacidad de enrutar paquetes, o lo que es lo mismo, se comporta como un router.

- **nat.** En esta tabla se producen los procesos de ***NAT*** que se configuren mediante ***iptables.*** Contiene tres cadenas predefinidas:
  - PREROUTING. Cadena donde se realiza el ***DNAT (Destination NAT).*** Por aquí pasarán los paquetes que entren en el sistema configurado con ***iptables.*** En este punto se decide a dónde se redirigen dichos paquetes. Coloquialmente hablando, aquí es donde se abren los puertos. Los paquetes pasan por esta cadena antes de la decisión de enrutado.
  - POSTROUTING. En esta cadena pasan los paquetes después de la decisión de enrutado. Se usa principalmente para hacer ***SNAT*** (***Source NAT),*** es decir, enmascarar los paquetes con la IP de la interfaz de salida. Ejemplo: transformación de IPs locales en IP pública.
  - OUTPUT. Cadena usada para realizar operaciones de ***NATeo*** para paquetes generados por un proceso local de la máquina configurada con ***iptables.***
- **mangle.** Es una tabla en la que se pueden manipular determinados aspectos de los paquetes. Entre ellos se destaca la modificación del ***TimeToLive***, ***TypeOfService*** y el marcado de paquetes. Este último proceso establece una marca a cada paquete a nivel de kemel. Dichas marcas pueden ser usadas para tomar decisiones de enrutado dependiendo de tipos de tráfico, etc. Util para balanceo de carga, implementaciones de ***QoS*** con ***qdisc*** configurables, etc. Las cadenas que posee la tabla mangle son:
  - PREROUTING
  - INPUT
  - FORWARD
  - OUTPUT
  - POSTROUTING
- **raw.** Se trata de una tabla relativamente nueva en la que se suele establecer una marca ***(NOTRACK)*** para evitar que ***netfilter*** realice un seguimiento del paquete. Se evitará que ***netfilter*** aplique ***conntrack*** al paquete. Posteriormente, mediante ejemplos, se verá para qué se usa ***conntrack.*** Las cadenas de las que dispone la tabla ***raw*** son:
  - PREROUTING
  - OUTPUT

Si se desea personalizar aún más el funcionamiento de ***iptables***, aparte de crear nuevas tablas como se comentó con anterioridad, es posible crear nuevas cadenas y enlazar las decisiones de sus reglas con otras cadenas.

A continuación se continuará explicando el funcionamiento y los parámetros disponibles en ***iptables*** mediante ejemplos. Entre otros aspectos se mostrarán las acciones o destinos disponibles para cuando un paquete coincida con una regla.

La siguiente imagen muestra un diagrama de flujo correspondiente a los diferentes estados con los que trabaja ***iptables.***

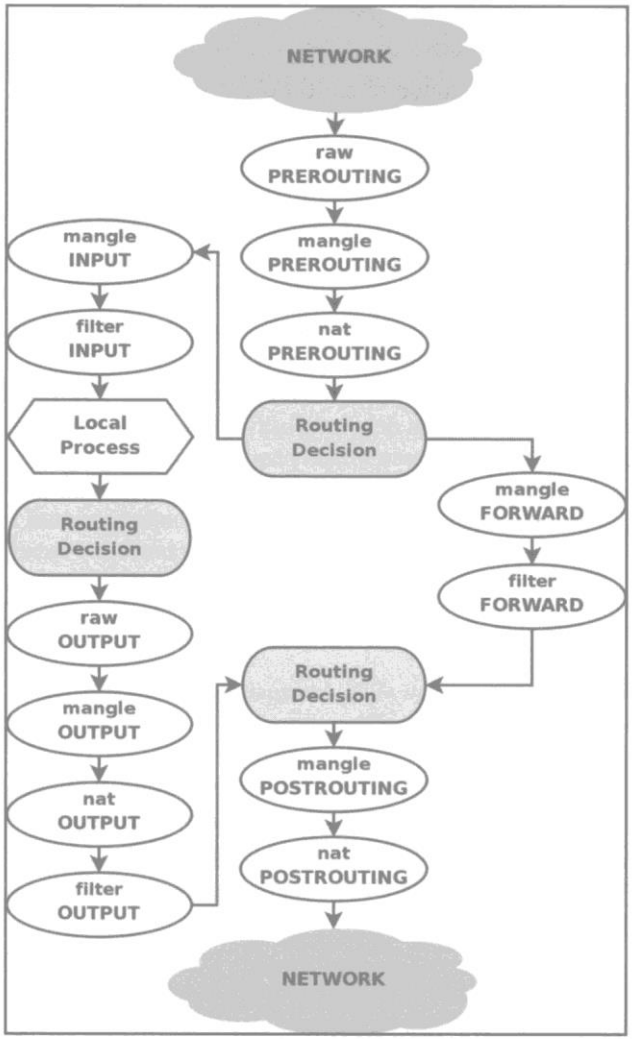


Fig 3.01: Esquema interno de *iptables*.

Ahora es el momento de ver cómo se crean y se eliminan reglas. También se verá cómo aplicar políticas por defecto para las cadenas.

Agregando reglas con iptables

Ahora es el momento de ver cómo se crean y se eliminan reglas. También se verá cómo aplicar políticas por defecto para las cadenas. La forma genérica de hacerlo es la siguiente:

iptables -t <nombreTabla> -A <cadena> <opciones> -j <acción>

- ***iptables*** es el comando en sí.
- El parámetro "-t" se utiliza para indicar la tabla con la que se está trabajando en el momento de lanzar el comando ***iptables.***
- El parámetro "-A" añade ('***append***') una regla a la cadena indicada.
- Donde <opciones> se pueden establecer diversos filtros para tratar de hacer coincidir un paquete por protocolo (***tcp***, ***udp***, ***icmp***, etc.), por puertos, ***IPs***, etc.
- Con "-j" se indica la acción que se le aplica a un paquete en caso de coincidir con la regla.

Ejemplos

• Descartar la navegación web (solo tráfico HTTP) por parte de las máquinas de la red local que pasen por la máquina que ejecuta ***iptables*** (entrando por la interfaz interna "ethl" y saliendo por la interfaz externa "ethO"):

```
# iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 80 -j DROP
```

Se puede observar cómo para identificar el tráfico HTTP se ha usado la opción "-p tcp" para indicar el tipo de protocolo, así como una sub-opción denominada "—dport" para indicar el puerto destino ("destination port").

• Se realiza el mismo proceso pero a la inversa. Denegamos la respuesta de servidores ***HTTP.***

```
# iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp --sport 80 -j DROP
```

En el caso de no especificar ninguna dirección origen o destino, como ocurre en este ejemplo y el anterior, se engloban todas las direcciones origen y destino.

• Se pasa a bloquear el tráfico ***SSH***, por lo que las máquinas de la red local no podrán conectar con un servidor ***SSH*** externo. Denegamos tanto la petición como la respuesta.

```
# iptables -t filter -A FORWARD -i eth1 -o eth0 -p tcp --dport 22 -j DROP
# iptables -t filter -A FORWARD -i eth0 -o eth1 -p tcp -sport 22 -j DROP
```

Se puede observar cómo en la regla de respuesta de SSH se establece el parámetro "—sport" para indicar el puerto origen ***("source port").***

• Ahora se procede a realizar SNAT para que los paquetes salientes de la máquina, ya sean generados localmente o re-enviados desde otra subred, salgan con la IP pública externa.

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Se ha indicado que a todos los paquetes que vayan a salir por la interfaz externa, ***ethO*** en este caso, se les aplique SNAT. Así, el paquete se podrá enrutar por Internet sin problemas.

Por defecto la política de las cadenas en la tabla filter es ACCEPT. Es por eso que se han aplicado dos pares de reglas (HTTP y SSH) para tráfico de la red local con acción DROP. No tendría sentido con la política ACCEPT establecer ACCEPT como acción de las reglas. Como norma

general las acciones establecidas serán la inversa de la que se realice con la política por defecto establecida.

Listando reglas con iptables

Si se desea obtener una visión de las reglas que están operativas en el sistema se puede realizar con las siguientes variantes del comando ***iptables***:

```
# iptables -t filter -nvL
```

El comando anterior muestra las reglas aplicadas para la tabla ***filter.*** En caso de omitir el parámetro ***-t*** se mostrará por defecto la tabla ***filter.***

```
# iptables -t nat -nvL
```

Se realiza exactamente lo mismo pero con la tabla ***nat.*** En caso de necesitar obtener los números asociados a cada regla, puede realizarse con el siguiente comando:

```
# iptables -t nat -nvL --line-numbers
```

Lo anterior resulta interesante para cuando se necesita eliminar una regla en concreto. Se obtiene el número de la regla de cada cadena y se procede a eliminar dicha regla utilizando el número asociado.

Eliminando reglas aplicadas

Cada vez que necesita restablecerse la configuración del ***firewall*** o simplemente modificar el comportamiento del mismo, será necesario eliminar reglas. A continuación se exponen un par de métodos para llevar a cabo dicha tarea.

Método selectivo

Cuando se desea eliminar una regla determinada de ***iptables***, previamente es necesario realizar una visualización de las reglas que están aplicadas en ese instante junto con sus números de regla. Puede obtenerse con el último comando del anterior apartado.

Si por ejemplo se quisiera eliminar la primera regla de la cadena ***POSTROUTING*** de la tabla ***nat*** habría que realizar algo como lo siguiente:

```
# iptables -t nat -D POSTROUTING 1
```

Como siempre, se indica la tabla sobre la que se desee realizar la operación con el parámetro ***-t.*** Acto seguido se indica de qué cadena se va a eliminar una regla con el parámetro ***-D*** (***delete***) seguido del nombre de la cadena y del número de la regla a eliminar.

Limpieza o 'flush'

Es probable que en caso de haberse realizado pruebas con ***iptables*** puedan quedar residuos de reglas antiguas que interfieran con las que se prueben más adelante; o simplemente que se desee restablecer el estado inicial desconfigurado de ***iptables.*** Los siguientes comandos eliminan todas las reglas de las tablas ***nat*** y ***filter.***

```
# iptables -t nat -F

# iptables -t filter -F
```

El parámetro ***-F ("flash")*** dejará limpias las tablas que se hayan indicado. Cabe destacar que al limpiar las tablas y cadenas de ***iptables*** no se restablece la política por defecto ***ACCEPT.*** Habrá que establecerla en el estado deseado con el comando que se explica a continuación.

Cambiando política por defecto

Para modificar el comportamiento por defecto de ***iptables*** en ciertas cadenas se utiliza el comando ***iptables*** con los siguientes parámetros:

iptables -t <tables> -P <cadena> <POLÍTICA>

Como ejemplo se pasa a establecer la política por defecto de la cadena ***INPUT***, de la tabla ***filter***, a ***DROP***:

```
iptables -t filter -P INPUT DROP
```

Es decir, se está descartando todo el tráfico entrante que vaya destinado a la máquina que configura ***iptables.***

Haciendo las reglas permanentes

Cuando la máquina en la que se configura ***iptables*** se reinicia, las reglas de ***iptables*** que se definieron desaparecen en el arranque. Volverán a existir las tablas, cadenas y políticas por defecto sin ninguna regla.

Existen varios métodos para hacer que las reglas de ***iptables*** se mantengan después de un reinicio. En este caso se muestra una muy clásica que consiste en introducir todos los comandos ***iptables*** que se deseen para configurar el entorno en un ***script en bash.*** Dicho script se ejecutará en cada arranque de la máquina lanzándolo por ejemplo lanzándolo desde ***/etc/rc.local.***

También es posible realizar esta tarea usando los comandos ***iptables-save***, ***iptables-restore*** e ***iptables-apply.***

Firewall de '2 patas'

Ahora que ya se ha explicado cómo crear reglas, cambiar políticas, etc. se procede a montar el siguiente entorno.

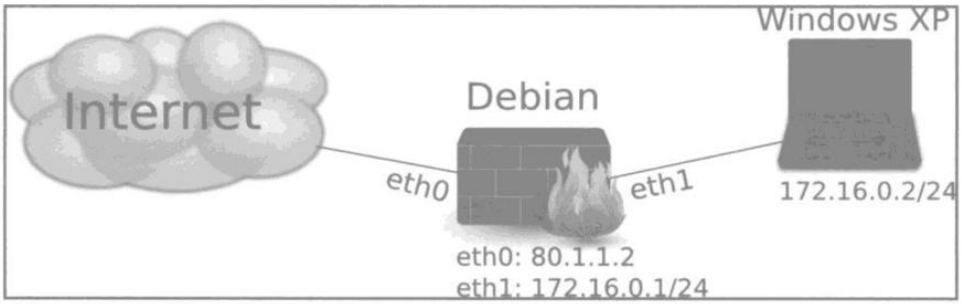


Fig 3.02: Esquema que representa una configuración de *firewall* de '2 patas'.

Se trata de un ***firewall*** de dos patas en el que se da conectividad hacia Internet a una máquina cliente, ***Windows XP*** en este ejemplo. La máquina que actúa como enrutador y ***firewall*** está ejecutando una ***Debían 6 'Squeeze*** \

Para el ejemplo se supone que la máquina que ejecuta ***iptables*** tiene asociada en su interfaz externa con/ZV/ pública 80.1.1.2 y en la interna la ***IP*** 172.16.0.1/24.

Por su parte, la máquina cliente tendrá asignada de forma manual una dirección IP que esté en la misma subred que su puerta de enlace. Para el ejemplo la 172.16.0.2/24. ¿Y cuál es su puerta de enlace? - En efecto, es la dirección ***IP*** de la interfaz interna de la máquina ***Debían.***

Un ejemplo funcional y sencillo, pero inseguro

Se va a permitir todo el tráfico desde la máquina ***Windows XP***, con lo que tendrá salida total a Internet. Los pasos que hay que seguir para tal finalidad son:

1. Convertir la máquina Debian en un router.
2. Hacer SNAT para los paquetes salientes por la interfaz externa de Debian.
3. Nada más. Hay que recordar que la política por defecto de las cadenas en ***iptables esACCEPT***, por lo que los paquetes pasarán limpiamente.

Se siguen los pasos citados arriba. Hay que empezar por activar el ***"bit de routing***" de la máquina ***Debian.***

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Con lo anterior se ha configurado la máquina como un router, pero ha de tenerse en cuenta que esa reglas es volátil en el momento que la máquina se apague o reinicie. Para que la máquina se

comporte como un router de forma permanente, sin tener que ejecutar la orden anterior en cada inicio, se puede establecer la siguiente directiva en el fichero ***/etc/sysctl.conf.***

```
net.ipv4.ip_forward=1
```

Por defecto la línea anterior está comentada, así que tan solo será necesario "descomentar" y salvar el fichero.

Siguiendo los pasos a realizar, llega el tumo de ***SNAT*** para enmascarar los paquetes que salgan a internet. Es decir, se re-escribe la cabecera de dichos paquetes estableciendo como ***IP*** origen la ***IP*** de la interfaz extema. Puede realizarse mediante dos métodos en caso de conocer la ***IP*** de la interfaz externa o que esta sea estática respectivamente.

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 80.1.1.2
```

Cualquiera de las dos reglas anteriores es correcta si la dirección ***IP*** es estática y es la 80.1.1.2. En el caso de disponer de ***IP*** dinámica se usa ***MASQUERADE***, que internamente averigua la ***IP*** asociada a la interfaz de la regla.

Si se prueba por ejemplo lanzar un***ping*** desde la máquina ***Windows XP*** hacia internet... ¡Funciona! Ya se ha conseguido hacer un router con dos simples reglas.

Pero tal y como está configurado el entorno en estos instantes se puede acceder de forma directa a todos los servicios de la máquina ***Debían*** que actúa como ***firewall.*** Todas las cadenas de la tabla ***filter*** están configuradas con política por defecto ***ACCEPT*** y no existe ninguna regla que descarte paquetes.

Aplicando política por defecto DROP

El ejemplo anterior está bien como prueba de que ***Debían*** es capaz de enrutar y hacer ***SNAT.*** Atendiendo a la seguridad no es en absoluto lo mejor que se puede hacer.

Es por ello que el entorno anteriormente planteado pasa a configurarse con política por defecto ***DROP*** en las cadenas de ***filter.*** Como consecuencia de ello es necesario crear reglas para permitir algún tipo de tráfico. El resumen de las reglas a aplicar es el siguiente:

4. Se permitirá el tráfico de ***loopback*** en la máquina ***Debían.***
5. La máquina ***Debían*** podrá realizar conexiones ***SSH.***
6. La máquina ***Debían*** es un servidor ***SSH.***
7. La máquina ***Debían*** tendrá permitido solo el tráfico ***ICMP*** saliente.
8. La máquina ***Debían*** podrá usar como cliente: ***DNS, HTTP*** y ***HTTPS.***
9. La máquina ***Windows XP*** dispone de un servidor web. Sólo tráfico ***HTTP.***

10. La máquina ***Windows XP*** es cliente ***DNS, HTTP*** y ***HTTPS.***
11. La máquina ***Windows XP*** tiene permitido el tráfico ***ICMP.***

A continuación se muestra como quedarían todas las condiciones anteriores en un script en ***bash.*** Se hace uso de variables en ***bash.***

```
#!/bin/bash

# Borrando reglas anteriores
iptables -F
iptables -t nat -F

# Estableciendo política por defecto a DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Se activa el bit de enrutado de la máquina
echo 1 > /proc/sys/net/ipv4/ip_forward

# Constantes y variables
LAN="172.16.0.0/24"
IFLAN="eth1"
IFEXT="eth0"

# ----------------------------------------
# Tráfico entrante y saliente de la máquina
# ----------------------------------------

# Se permite todo el tráfico loopback
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Se permiten consultas DNS hacia internet
iptables -A INPUT -i $IFEXT -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p udp --dport 53 -j ACCEPT
```

```
# Se permiten consultas HTTP y HTTPS hacia internet
iptables -A INPUT -i $IFEXT -p tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -i $IFEXT -p tcp --sport 443 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p tcp --dport 443 -j ACCEPT

# Se permiten consultas DHCP
iptables -A INPUT -i $IFEXT -p udp --sport 67:68 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p udp --dport 67:68 -j ACCEPT

# Se permite solo el tráfico ICMP saliente y su respuesta
iptables -A INPUT -i $IFEXT -p icmp -m state --state ESTABLISHED,RELATED \
-j ACCEPT
iptables -A OUTPUT -o $IFEXT -p icmp -j ACCEPT

# Se permite el tráfico SSH por la interfaz externa
iptables -A INPUT -i $IFEXT -p tcp --sport 22 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p tcp --dport 22 -j ACCEPT

# La máquina es servidor SSH (se acepta SSH por cualquier IF)
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT


# ---------------
# Tráfico FORWARD
# ---------------

# Se permite el tráfico ICMP
iptables -A FORWARD -p icmp -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p icmp -o $IFLAN -i $IFEXT -j ACCEPT

# Se permite tráfico HTTP y HTTPS
```

```
iptables -A FORWARD -p tcp --dport 80 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 80 -o $IFLAN -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 443 -o $IFLAN -i $IFEXT -j ACCEPT

# La máquina es servidor HTTP
iptables -A FORWARD -p tcp --sport 80 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 80 -o $IFLAN -i $IFEXT -j ACCEPT

# Se permiten consultas DNS
iptables -A FORWARD -p udp --dport 53 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -o $IFLAN -i $IFEXT -j ACCEPT

# ---
# NAT
# ---

# Se hace DNAT (se abre un puerto) para Windows XP. Puerto 80
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 80 -j DNAT \
--to-destination 172.16.0.2

# Se hace SNAT para la red interna
iptables -t nat -A POSTROUTING -s $LAN -o $IFEXT -j MASQUERADE
```

Ejecutando el script anterior, que podría estar contenido en el fichero ***/usr/local/sbin*** por ejemplo, se obtiene un entorno de ***firewall*** de dos patas con las reglas que se definieron anteriormente.

Por ejemplo si desde Internet se intentase acceder a la ***IP*** pública 80.1.1.2 del ***firewall*** mediante un navegador web, se estaría accediendo al servidor web de la máquina ***Windows XP.*** Se redirigen las peticiones con puerto destino 80/tcp que entren por el ***firewall*** a la máquina ***Windows XP.*** No solo hay que redirigir la conexiones mediante la tabla ***nat***, también es necesario aceptar el tráfico entrante con la tabla ***filter.*** Para entender correctamente el proceso, es conveniente observar el flujo de las conexiones con el diagrama de ***iptables*** del inicio de este capítulo.

La política ***DROP*** desecha los paquetes que la máquina se envía a sí misma, al igual que cuando los recibe de sí misma. Es por ello que se acepta todo el tráfico entrante y saliente de la interfaz ***loopback.*** Siendo un poco paranoicos podría controlarse el tráfico entrante y saliente de la interfaz ***loopback.***

Por otra parte, se aprecia que no se han especificado hosts en las reglas de filtrado, sino que directamente se han usado las interfaces. En un entorno en producción debería utilizarse reglas con subredes/hosts e interfaces y no solo con las interfaces. Para tal cometido se utilizan los parámetros ***-d*** y ***-s.*** Se especifican así el destino y origen respectivamente.

Firewall de '3 patas'

En el ejemplo anterior se ha establecido un entorno equiparable al que puede encontrarse en cualquier hogar con conexión a Internet. Es decir, existen máquinas interconectadas mediante un router doméstico (modem, router, switch y AP habitualmente) con salida hacia Internet. En dicho router se define una red interna, se crean redirecciones a máquinas internas (***NAT***), etc.

Para entornos con servidores que deben estar accesibles desde Internet, así como desde la red interna, suelen implementarse ***firewall*** un poco más avanzados. Existen muchos tipos, pero en esta ocasión se opta por implantar un ***firewall*** de tipo ***3 patas*** por el hecho de ser no excesivamente complejo de configurar.

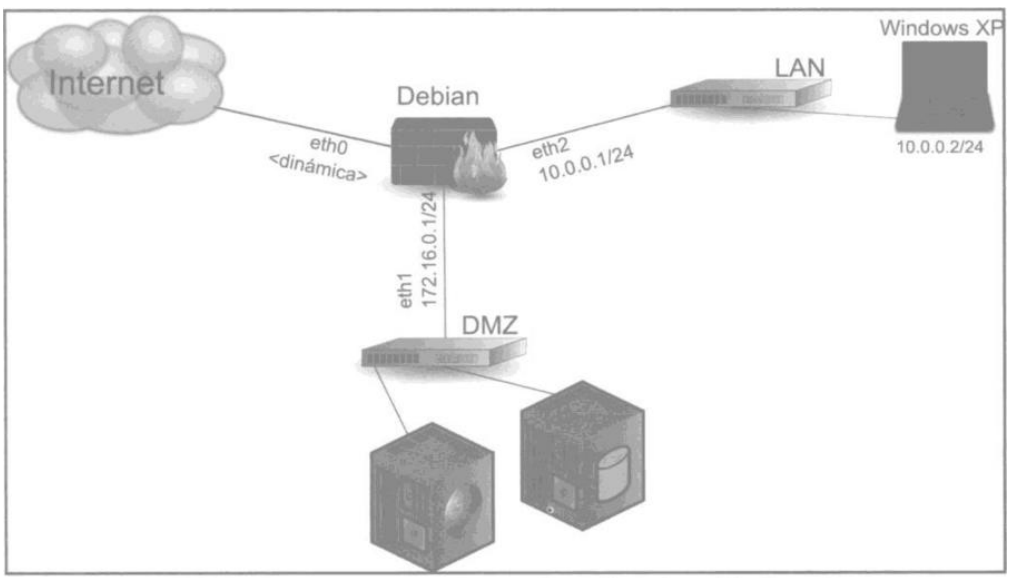


Fig 3.03: Esquema que representa una configuración de *firewall* de '3 patas'.

El entorno que se implementa en este caso es el mostrado en la imagen anterior. Sus características son las siguientes:

- Política por defecto ***DROP.***
- Se permitirá el tráfico de ***loopback*** en la máquina ***firewall***.
- La máquina ***firewall*** podrá realizar conexiones ***SSH*** hacia Internet.

- La máquina ***firewall*** es un servidor ***SSH.***
  - Accesible desde la LAN.
  - Accesible desde las direcciones IP 8.8.8.8 y 8.8.4.4 de Internet.
- La máquina ***firewall*** tendrá permitido el tráfico ***ICMP.*** Responderá por todas las interfaces a dicho tráfico.
- Desde la máquina ***firewall*** se podrán iniciar conexiones ***SSH*** hacia los servidores de la DMZ.
- La máquina ***firewall*** tiene conectividad como cliente a recursos ***HTTP***, ***HTTPS*** y ***DNS.***
- La red DMZ tendrá acceso como cliente a los recursos ***HTTP***, ***HTTPS*** y ***DNS.***
- Desde la red DMZ se acepta el tráfico ***ICMP*** saliente de tipo 0, 3, 8 y 11.
- Desde la red DMZ se ofrecerán los siguientes servicios:
  - Servidor web en la IP 172.16.0.2. Se ofrecen webs seguras.
  - Servidor de correo en la máquina 172.16.0.3. ***SMTP***, ***SMTPS***, ***POP3, POP3S***, ***IMAP3*** e ***IMAPS.***
- Desde la red LAN se permitirá el acceso a recursos ***HTTP/S***, ***DNS*** y ***FTP.***
- Desde la red LAN se acepta el tráfico ***ICMP*** saliente de tipo 0, 3, 8 y 11.

Como ocurría en el ejemplo del ***firewall*** de dos patas, se muestra a modo de script cómo quedarían definidas las reglas anteriores mediante ***iptables.***

```
#!/bin/bash

# Borrando reglas anteriores
iptables -F
iptables -t nat -F

# Estableciendo política por defecto a DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

# Se activa el bit de enrutado de la máquina
echo 1 > /proc/sys/net/ipv4/ip_forward

# Constantes y variables
LAN="10.0.0.0/24"
```

```
DMZ="172.16.0.0/24"
IFLAN="eth2"
IFDMZ="eth1"
IFEXT="eth0"

# Aquí podrían establecerse reglas adicionales para aumentar la
# seguridad del entorno.

# ----------------------------------------
# Tráfico entrante y saliente de la máquina
# ----------------------------------------

# Se permite todo el tráfico loopback
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Se permiten consultas DNS hacia internet
iptables -A INPUT -i $IFEXT -p udp --sport 53 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p udp --dport 53 -j ACCEPT

# Se permiten consultas HTTP y HTTPS hacia internet
iptables -A INPUT -i $IFEXT -p tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -i $IFEXT -p tcp --sport 443 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p tcp --dport 443 -j ACCEPT

# Se permite tráfico ICMP por todas las interfaces
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT

# Se permite el tráfico SSH hacia Internet por la interfaz externa
iptables -A INPUT -i $IFEXT -p tcp --sport 22 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -p tcp --dport 22 -j ACCEPT
```

```
# Se permite el tráfico SSH hacia la DMZ
iptables -A INPUT -i $IFDMZ -s $DMZ -p tcp --sport 22 -j ACCEPT
iptables -A OUTPUT -o $IFEXT -d $DMZ -p tcp --dport 22 -j ACCEPT

# La máquina es servidor SSH (se acepta SSH por cualquier IF)
iptables -A INPUT –i $IFLAN -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT –o $IFLAN -p tcp --sport 22 -j ACCEPT
iptables -A INPUT –i $IFEXT -s 8.8.8.8 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT –o $IFEXT -d 8.8.8.8 -p tcp --sport 22 -j ACCEPT
iptables -A INPUT –i $IFEXT -s 8.8.4.4 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT –o $IFEXT -d 8.8.4.4 -p tcp --sport 22 -j ACCEPT


# ---------------
# Tráfico FORWARD
# ---------------

# ------DMZ------
# Se permite todo tipo de tráfico ICMP con origen DMZ y solo alguno
# con destino DMZ
iptables -A FORWARD -i $IFDMZ -s $DMZ -p icmp -j ACCEPT
iptables -A FORWARD -o $IFDMZ -d $DMZ -p icmp --icmp-type \
destination-unreachable -j ACCEPT
iptables -A FORWARD -o $IFDMZ -d $DMZ -p icmp --icmp-type time-exceeded -j ACCEPT
iptables -A FORWARD -o $IFDMZ -d $DMZ -p icmp --icmp-type echo-reply -j ACCEPT
iptables -A FORWARD -o $IFDMZ -d $DMZ -p icmp --icmp-type echo-request -j ACCEPT

# Se permite tráfico HTTP y HTTPS
iptables -A FORWARD -p tcp --dport 80 -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 80 -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 443 -o $IFDMZ -i $IFEXT -j ACCEPT

# Se permiten consultas DNS
```

```
iptables -A FORWARD -p udp --dport 53 -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -o $IFDMZ -i $IFEXT -j ACCEPT

# La IP 172.16.0.2 es servidor web (http, https)
iptables -A FORWARD -p tcp --sport 80 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 80 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 443 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT

# La IP 172.16.0.3 es servidor de correo
iptables -A FORWARD -p tcp --sport 25 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 25 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 465 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 465 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 110 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 110 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 995 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 995 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 220 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 220 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 993 -s $DMZ -i $IFDMZ -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 993 -d $DMZ -o $IFDMZ -i $IFEXT -j ACCEPT

# ------LAN------
# Se permite todo tipo de tráfico ICMP con origen LAN y solo alguno
# con destino LAN
iptables -A FORWARD -i $IFLAN -s $LAN -p icmp -j ACCEPT
iptables -A FORWARD -o $IFLAN -d $LAN -p icmp --icmp-type \
destination-unreachable -j ACCEPT
iptables -A FORWARD -o $IFLAN -d $LAN -p icmp --icmp-type \
time-exceeded -j ACCEPT
iptables -A FORWARD -o $IFLAN -d $LAN -p icmp --icmp-type \
echo-reply -j ACCEPT
iptables -A FORWARD -o $IFLAN -d $LAN -p icmp --icmp-type \
```

```
echo-request -j ACCEPT

# Se permite tráfico HTTP y HTTPS
iptables -A FORWARD -p tcp --dport 80 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 80 -o $IFLAN -i $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 443 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --sport 443 -o $IFLAN -i $IFEXT -j ACCEPT

# La máquina es servidor HTTP
iptables -A FORWARD -p tcp --sport 80 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p tcp --dport 80 -o $IFLAN -i $IFEXT -j ACCEPT

# Se permiten consultas DNS
iptables -A FORWARD -p udp --dport 53 -i $IFLAN -o $IFEXT -j ACCEPT
iptables -A FORWARD -p udp --sport 53 -o $IFLAN -i $IFEXT -j ACCEPT

# ---
# NAT
# ---

# Se hace DNAT ('se abren' puertos)
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 80 -j DNAT \
--to-destination 172.16.0.2
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 443 -j DNAT \
--to-destination 172.16.0.2

iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 25 -j DNAT \
--to-destination 172.16.0.3
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 465 -j DNAT \
--to-destination 172.16.0.3
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 110 -j DNAT \
--to-destination 172.16.0.3
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 995 -j DNAT \
--to-destination 172.16.0.3
```

```
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 220 -j DNAT \
--to-destination 172.16.0.3
iptables -t nat -A PREROUTING -i $IFEXT -p tcp --dport 993 -j DNAT \
--to-destination 172.16.0.3

# Se hace SNAT para las rede internas, DMZ y LAN.
iptables -t nat -A POSTROUTING -s $DMZ -o $IFEXT -j MASQUERADE
iptables -t nat -A POSTROUTING -s $LAN -o $IFEXT -j MASQUERADE
```

Con el script anterior se implementa un simple ***firewall*** de tres patas. Se distinguen claramente las secciones para el tráfico de E/S de la propia máquina, ***INPUT*** y ***OUTPUT***; así como el tráfico no dirigido a ella misma, ***FORWARD.***

Por último, se establecen las reglas para NAT. Por una parte, el caso de ***DNAT***' donde determinadas conexiones que llegan al ***firewall*** son dirigidas hacia otras máquinas de la red interna, o lo que vulgarmente es conocido como abrir un puerto. Finalmente se incluyen un par de reglas para realizar ***SNAT*** a las conexiones salientes por la interfaz externa, con IP pública, del ***firewall*** con origen las redes internas DMZ y LAN. Como se explicó con anterioridad, se enmascaran las direcciones IP privadas con la IP pública de la interfaz de salida.

Una de las opciones más interesantes de las que se dispone en ***netfilter,*** y como consecuencia su manejo desde ***iptables***, es el seguimiento de las conexiones. Todo ello es posible gracias a ***Netfilter Conntrack.*** Es pues posible dotar al ***firewall*** de un control sobre los estados de las conexiones e increíblemente, es aplicable también al protocolo UDP y no solo a TCP como era de esperar. Supone, aún más, un grado de control sobre las conexiones que se permiten mediante las reglas del cortafuegos, sin embargo no se han establecido dichas directivas para no complicar en exceso este ejemplo de cortafuegos de 3 patas.

Protecciones adicionales

A pesar de que el ***firewall*** está configurado con política por defecto ***DROP*** tanto en ***INPUT***, ***OUTPUT y FORWARD,*** aún es posible proteger un poco más el entorno. Para ello se muestran, en el siguiente fragmento de script, algunas de las muchas medidas de protección posibles. Estas reglas deben estar colocadas justo antes de las reglas para conexiones permitidas.

```
# -----------------
# Reglas especiales
# -----------------
# -- INPUT --
# Restringir determinados tipos de escaneo y/o paquetes mal formados
```

```
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags ACK ACK -m state \
--state NEW -j REJECT
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags RST RST -m state \
--state NEW -j REJECT
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags PSH PSH -m state \
--state NEW -j REJECT
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags FIN FIN -m state \
--state INVALID -j REJECT
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags FIN,PSH,URG \
FIN,PSH,URG -j DROP
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags ALL \
ACK,RST,SYN,FIN -j DROP
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A INPUT -i $IFEXT -p tcp --tcp-flags ALL NONE -j DROP

# Limitar conexiones de tipo TCP SYN entrantes
iptables -A INPUT -i $IFEXT -p tcp --syn -m recent --set
iptables -A INPUT -i $IFEXT -p tcp --syn -m recent --update \
--seconds 5 --hitcount 20 –j DROP

# Limitar a un ping por segundo y dirección IP
iptables -A INPUT -i $IFEXT -p icmp --icmp-type echo-request \
-m hashlimit --hashlimit-name ping --hashlimit-above 1/s \
--hashlimit-burst 2 --hashlimit-mode srcip -j REJECT

# -- FORWARD
# Restringir determinados tipos de escaneo y paquetes mal formados
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags ACK ACK -m state \
--state NEW -j REJECT
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags RST RST -m state \
--state NEW -j REJECT
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags PSH PSH -m state \
--state NEW -j REJECT
```

```
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags FIN FIN -m state \
--state INVALID -j REJECT
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags FIN,PSH,URG \
FIN,PSH,URG -j DROP
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags ALL \
ACK,RST,SYN,FIN -j DROP
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -A FORWARD -i $IFEXT -p tcp --tcp-flags ALL NONE -j DROP

# Limitar conexiones de tipo TCP SYN entrantes
iptables -A FORWARD -i $IFEXT -p tcp --syn -m recent --set
iptables -A FORWARD -i $IFEXT -p tcp --syn -m recent --update \
--seconds 5 --hitcount 20 -j DROP

# Limitar a un ping por segundo y dirección IP
iptables -A FORWARD -i $IFEXT -p icmp --icmp-type icmp-request \
-m hashlimit --hashlimit-name ping --hashlimit-above 1/s \
--hashlimit-burst 2 --hashlimit-mode srcip -j REJECT
```

En el ejemplo anterior pueden observarse algunos de los parámetros de ***iptables*** que hacen uso de ***CONNTRÁCK.*** En este caso se han usado para identificar paquetes con estados ***NEW*** e ***INVALID*** con el fin de detectar escáneres de puertos que usen métodos ***FIN*** y ***ACK,*** y como consecuencia denegarlos.

Otra característica interesante que se ha implementado mediante las reglas anteriores ha sido la capacidad de limitar las conexiones que pasan o entran por el ***firewall.*** En concreto se han limitado las solicitudes de ***ping*** desde una IP a una velocidad de una petición por segundo. De ese modo se está evitando que una determinada IP pueda realizar un ataque ***ICMP flood*** mediante el uso de ***ping.*** Ha sido posible usando el módulo ***hashlimit*** de ***iptables. Hashlimit*** permite establecer un ratio de conexiones, una ráfaga, así como establecer un modo de trabajo entre otras opciones. En este ejemplo los parámetros se han establecido respectivamente como 1/s (uno por segundo), 2 y ***srcip.*** El destino de los paquetes que superen esos límites será ***REJECT.***

Para evitar ataques de tipo ***SYN flood*** se ha establecido un mecanismo para descartar conexiones entrantes de direcciones que superen más de 20 paquetes ***TCP*** tipo ***SYN*** en 5 segundos. Todo ello es posible usando el módulo ***recent*** de ***iptables.***

Todos los métodos anteriores han sido aplicados tanto a la cadena ***INPUT*** como ***FORWARD*** de la tabla ***FILTER.*** Es decir, una vez que se haya tomado la decisión de enrutamiento, se aplican los filtros a los paquetes destinados al propio ***firewall***, así como a otras máquinas. En ambos casos, se aplica a los paquetes que lleguen por la interfaz externa.

Obviamente, todas las configuraciones que limiten el número de conexiones deben estudiarse en profundidad y comprender sus consecuencias antes de aplicarlas. Dependiendo de la carga a la que esté sometida una red y por lo tanto su ***firewall***, puede que haya que incrementar algunos valores pues se estarán descartando "conexiones lícitas".

Front-ends para iptables

Se ha mostrado cómo se implementan***firewalls*** mediante el uso de reglas definidas con ***iptables.*** En concreto para entornos de dos y tres patas. Queda demostrada la potencia que tiene ***iptables*** para incrementar la seguridad en entornos de red mediante la combinación de sus reglas tradicionales, así como el uso de su gran variedad de módulos. A pesar de ello, se puede llegar a pensar que el uso de ***iptables*** puede resultar complicado, tedioso y lento. Es por lo que en esta sección se pretenden enumerar un par de ***front-ends*** para la configuración de ***firewalls*** en sistemas ***GNU/Linux.*** Usando, por supuesto, como ***back-end*** las herramientas proporcionadas por ***netfilter***; entre ellas ***iptables.***

UFW

Uncomplicated FireWall, UFW, es un ***front-end***para ***iptables*** para facilitar la configuración y manejo de ***firewall*** basado en ***Netfilter.*** Proporciona una interfaz de línea de comandos y está especialmente pensado para ***firewalls*** en máquinas finales.

La aplicación se instala mediante el paquete ***ujw*** en el caso de ***Debían*** y distribuciones basadas en ella. Una vez instalado, algunas de las operaciones que se pueden realizar con el comando ***ujw*** son las siguientes:

- ufw enable | disable | reload. Para habilitar, deshabilitar y recargar el ***firewall ujw.***
- ufw default allow | deny | reject [incoming | outgoing]. Se indica la política por defecto del ***firewall.*** En caso de especificar el último parámetro, en cualquiera de sus dos valores, se aplicará la política al tráfico entrante o saliente.
- ufw status. Se muestra el estado del ***firewall***, así como sus reglas.
- ufw [delete] allow|deny|reject|limit [in|out on INTERFACE] [log|log-all] [proto protocol] [from ADDRESS [port PORT]] [to ADDRESS [port PORT]]. Se añade o elimina, dependiendo de si se incluye o no la opción ***delete***, una regla con acción allow, deny, reject o limit. Además de indicar la interfaz, direcciones y/o puertos.

APF-FIREWALL

APF, Advanced Policy Firewall, es un cortafuegos basado en ***iptables.*** Los ficheros de configuración de ***APF*** están diseñados para ser sencillos y fáciles de seguir desde el inicio hasta el final. La

administración del ***firewall*** se puede llevar a cabo mediante la aplicación de línea de comandos ***apf*** que incluye información de uso y todas las características que se puedan esperar de una solución ***firewall.***

Resumen de las características:

- Ficheros de configuración detallados y bien documentados.
- Filtrado entrante y saliente separado.
- Filtrado salientes basado en ***uid*** de procesos.
- Filtrado de red basado en aplicaciones.
- Prevención de intrusiones mediante bloqueo de direcciones reactivo, ***RAB.***
- Filtrado por tipos de transportes y protocolos con diferentes acciones aplicables.
- Prevención de ataques mediante la limitación de paquetes.
- Integración con listas de bloqueo para redes de actividad sospechosa.
- Chequeo de paquetes para evitar que salgan o entren paquetes mal formados.

iptables e ipv6

Se ha descrito cómo es posible proteger una red ***ipv4*** mediante el uso de ***iptables*** para la implementación de ***firewalls.*** A pesar de que aún, grandes organizaciones así como la mayoría de las pequeñas y medianas empresas además de los hogares, utilizan ***ipv4***; lo cierto es que la tendencia está cambiando y es prácticamente necesario y obligatorio dar el salto a ***ipv6.*** Como no podía ser de otra modo todas las distribuciones ***Linux*** actuales tienen soporte completo para ivp6, en el caso de Debian desde la versión 6 ***Squeeze.***

En ipv6, más allá de la cantidad de direcciones disponibles existentes, se introdujeron numerosos cambios con respecto a ipv4 que lo hacen diferente en algunos aspectos con respecto a su antecesor. Algunos de ellos son:

- Autoconfiguración de interfaces con direcciones de enlace local mediante el uso de la dirección ***MAC*** del dispositivo configurado. El equivalente a las direcciones 169.254.0.0/16 en ***ipv4.***

  Sustitución de ***ARP*** por ***NDP***, ***Neigbor discovery protocol.***

- Desaparición del mecanismo ***NAT, NetWork address translation.*** No tenía sentido continuar utilizándolo ya que se introdujo en su momento como uno de los parches para subsanar la escasez de direcciones ipv4. Sin duda, se trata de una de las consecuencias más beneficiosas sobre todo atendiendo a las comunicaciones multimedia. El uso de ***NAT*** suele dar muchos quebraderos de cabeza en este aspecto.

En cuanto al uso de ***iptables*** para ***ipv6*** es necesario utilizar un comando o aplicación alternativa para la configuración de reglas. Se trata de ***ipótables***, tal y como ocurre con ***pingó*** o ***traceroute6.***

Su funcionamiento y uso continúa siendo como en ***ipv4***, salvo que entre algunos cambios el más destacable es la lógica desaparición de la tabla ***nat.***

Se ha de tener especial cuidado con el tratamiento del tráfico ICMPv6 pues es fundamental para el correcto funcionamiento de ipv6. Con él se permiten entre otros aspectos todo lo relacionado con ***NDP, Neighbor discovery protocol.***

Por otra parte, como se comentó al inicio de este punto, aún existen muchos entornos en los que se utiliza íntegramente ***ipv4.*** Es en este tipo de casos, y más aún si no se tiene previsión de cambio a corto plazo, donde se recomienda la protección total o desactivación del protocolo ***ipv6.*** Para denegar el tráfico se puede utilizar.

```
# ip6tables -P INPUT DROP
# ip6tables -P OUTPUT DROP
# ip6tables -P FORWARD DROP
```

Si por el contrario se desea desactivar el soporte de ipvó por completo, en Debian es posible agregar la siguiente línea en el fichero ***/etc/sysctl.conf.***

```
net.ipv6.conf.all.disable_ipv6=1
```

De ese modo en el próximo reinicio de la máquina el soporte ***ipv6*** estará completamente desactivado. Aunque también es posible desactivar al vuelo el protocolo ***ipv6*** ejecutando la siguiente orden en una terminal con privilegios de root:

```
echo "1" > /proc/sys/net/ipv6/conf/all/disable_ipv6
```

La orden anterior puede resultar un tanto agresiva si algunas aplicaciones o demonios están configurados para utilizar ***ipv6.*** De igual modo, para evitar problemas de funcionamiento en aplicaciones preparadas para ***ipv6***, es recomendable eliminar o comentar las siguientes líneas del fichero ***/etc/hosts.***

```
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Corresponden a las direcciones ***localhost*** y ***multicast*** en redes ***ipv6.***

2.VPN

Como ocurre con todo, los métodos y formas de trabajo van evolucionando con el paso del tiempo. Las necesidades cambian y cada vez son más comunes las empresas que promueven el teletrabajo o que cuentan con empleados que deben realizar trabajos a distancia en determinadas ocasiones, cubrimiento de diferentes franjas horarias a lo ancho del planeta, etc. Es por ello que se necesitan ofrecer ciertos mecanismos para que un empleado pueda acceder de forma remota al entorno de trabajo, sin comprometer su seguridad ni la de su empresa.

Anteriormente se ha descrito cómo proteger un entorno mediante la configuración de cortafuegos con ***iptables.*** Es solo un apartado más para tratar de securizar un entorno, pero no es suficiente para el caso que se está planteando. En este caso es cuando 'entran enjuego' las ***VPN.***

Definición y tipos

Virtual Prívate NetWork, ***VPN***, o ***Red Privada Virtual*** en castellano, es un concepto que describe cómo extender una red local a través de entornos abiertos y hostiles como puede serlo Internet. La idea consiste en proporcionar unos túneles virtuales que transportan el tráfico de red de forma segura.

A pesar de que la idea es sencilla, es decir conectar a una red interna desde una red externa, existen varios tipos de implementación para ***VPN:***

VPN Punto a punto. Se trata de un entorno que suele implantarse cuando se desean conectar, por ejemplo, sedes remotas de una empresa. Con el fin de abaratar costes se utiliza Internet como medio de comunicación. Para conseguir que la comunicación sea cifrada y exista privacidad, se usan diferentes implementaciones de ***VPN.***

- ***VPN*** de acceso remoto, también conocida como ***Road Warrior.*** Son entornos que se implementan cuando equipos externos deben conectarse a la red interna de una empresa. Usando una conexión a Internet el equipo remoto se conecta al servidor ***VPN*** de la compañía y este proporciona conectividad segura hacia los recursos internos. Este será el entorno que se implementará como ejemplo para mostrar dos implementaciones de ***VPN.***

Dependiendo de la implementación y tecnología usada para crear un entorno ***VPN*** se obtendrán una serie de beneficios como pueden ser los siguientes:

- Autentificación. La máquina o persona que se conecta a través de una ***VPN*** debe identificarse para usar los recursos, por lo que siempre se podrá saber qué o quién realiza determinados accesos o usos.
- Confidencialidad. Mediante el intercambio inicial de credenciales y claves de sesión se procede a cifrar la comunicación para que solo los extremos que intervienen en la ***VPN*** puedan descifrar los datos e interpretarlos convenientemente.

- Integridad. Mediante el uso de diferentes algoritmos de hash es posible verificar que los datos no han sido alterados. Se comprueba la integridad de dichos datos.
- No repudio. Al existir un proceso de autenticación, es lógico que todas las acciones que se realicen queden registradas y asociadas a un determinado usuario o máquina.

Existen diversas implementaciones para crear ***VPN.*** En este caso se han elegido ***PPTP*** dada su popularidad y ***OpenVPN.*** Se trata de dos métodos muy conocidos y extendidos en entornos ***GNU/ Linux.***

El entorno de pruebas ideado es como el que se aprecia en la siguiente imagen, una ***VPN*** de tipo ***Road Warrior.***

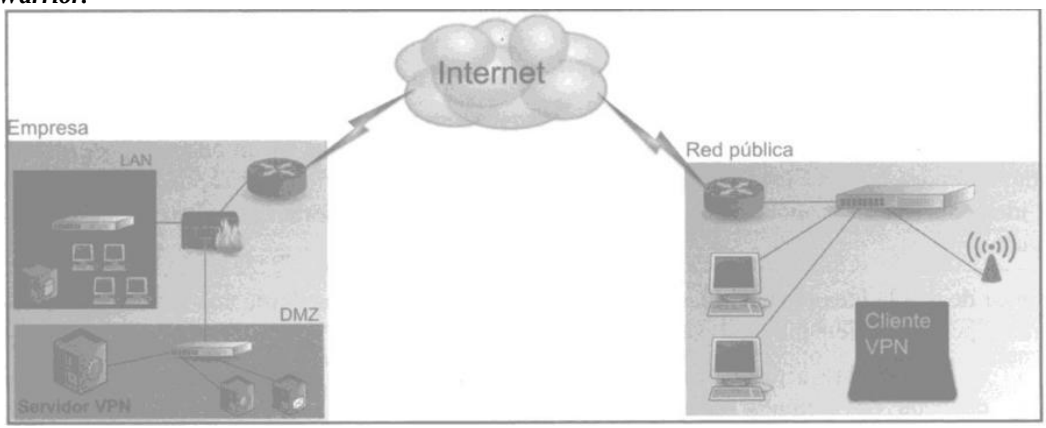


Fig 3.04: Esquema de una conexión VPN.

PPTP, Point-to-point Tunneling Protocol

PPTP, ***Point-to-Point Tunneling Protocol***, es un método para la implementación para la implementación de soluciones VPN. Su uso está muy extendido dada su facilidad de configuración, así como la amplia compatibilidad entre diferentes plataformas que abarcan desde dispositivos móviles, servidores, etc.

Para ***PPTP*** se utiliza una conexión TCP, escuchando el servidor en el puerto 1723. Dicha conexión es utilizada como inicialización del protocolo ***GRE***, ***Generic Routing Encapsulation***, destinado al enrutado del tráfico tunelizado. Los mecanismos anteriores permiten la creación de un túnel, pero no el cifrado ni autenticación del mismo. Para esta última tarea se utiliza el protocolo PPP, ***Point- to-Point Protocol.***

PPP ofrece diferentes mecanismos de autenticación. Los más destacados y utilizados son ***MSCHAPv2*** y ***EAP-TLS.*** El primero para clave precompartida o preshared key y el segundo utilizando una infraestructura ***PKI, Public Key Infraestructure.***

Instalación y configuración en Debían

Como se comentaba, una de las principales ventajas de este sistema es su sencilla configuración y su bajo consumo de recursos. Para instalar un servidor PPTP en una máquina Debian basta con ejecutar los siguientes comandos:

```
# apt-get update
# apt-get install pptpd
```

Como dependencias se instalarán los paquetes ***ppp*** y ***bcrelay.***

Una vez instalado, llega el tumo de la configuración y para ello deben modificarse varios archivos. En concreto son: ***/etc/pptpd.conj,\ /etc/ppp/pptpd-options*** y ***/etc/ppp/chap-secrets*** que contienen las siguientes configuraciones respectivamente.

En el fichero ***/etc/pptpd.conf*** se establece, entre otros parámetros, el direccionamiento IP de los clientes, así como el del propio servidor VPN.

```
option /etc/ppp/pptpd-options
logwtmp
localip 192.168.87.56
remoteip 192.168.87.215-220
```

En este segundo fichero se establecer opciones correspondientes a la parte de ***PPP***, tales como métodos de autenticación, rutas, ***DNS***, etc.

```
name pptpd
refuse-pap
refuse-chap
refuse-mschap
require-mschap-v2
require-mppe-128
ms-dns 192.168.87.1
proxyarp
nodefaultroute
lock
nobsdcomp
```

En el tercer fichero, ***/etc/ppp/chap-secrets***, se establecen las credenciales de acceso para los clientes. Es posible utilizar uno genérico o bien uno por cada cliente. El primer y tercer parámetro se corresponden con el nombre de usuario y el ***password*** respectivamente.

```
carlos  pptpd   W34k_P4ssw0rd!        *
```

Todos los valores están establecidos a los que están por defecto salvo las credenciales, los servidores DNS y el direccionamiento IP. Recargando la configuración del servidor ***pptpd***, la VPN estará lista para recibir conexiones de clientes.

A simple vista todo resulta muy sencillo de configurar y utilizar, pero ¿se está seguro utilizando una conexión VPN con clave pre-compartida utilizando ***MSCHAPv2*** o inferior? Como se verá a continuación, la respuesta es no.

Cracking de conexiones PPTP

Siempre que se ha usado una infraestructura VPN conformada por ***PPTP*** se ha recomendado utilizar el método de autenticación ***MSCHAPv2*** por ser el más seguro cuando se utiliza clave simétrica. El problema principal reside en que PPP no proporciona ningún canal cifrado mientras se está negociando la autenticación. Por lo tanto, es susceptible a ataques ***MITM***, ***Man in the Middle*** y su posterior explotación debido a que la autenticación se realiza mediante una contraseña o frase de paso potencialmente descifrable mediante ataques de fuerza bruta y diccionario.

Hasta hace poco se creía que utilizando una passphrase de gran fortaleza para ***MSCHAPv2*** se estaba seguro a pesar de disponer de realizar ataques offline tras capturar los tokens de autenticación mediante un ataque ***MITM.*** Todo esto se acabó cuando en la DEFCON20, el hacker Moxie Marlinspike presentó un método por el cual era capaz de extraer passwords ***MSCHAPv2*** mediante fuerza bruta, usando una infraestructura cloud, en veinticuatro horas aproximadamente.

Se puede concluir por lo tanto que ***MSCHAPv2*** está totalmente roto y no es seguro, por lo tanto con ***PPTP*** no se estaría seguro utilizando autenticación por ***password*** o passphrase. A continuación se muestra cómo de sencillo es interceptar un inicio de sesión para tratar de extraer la contraseña a posteriori.

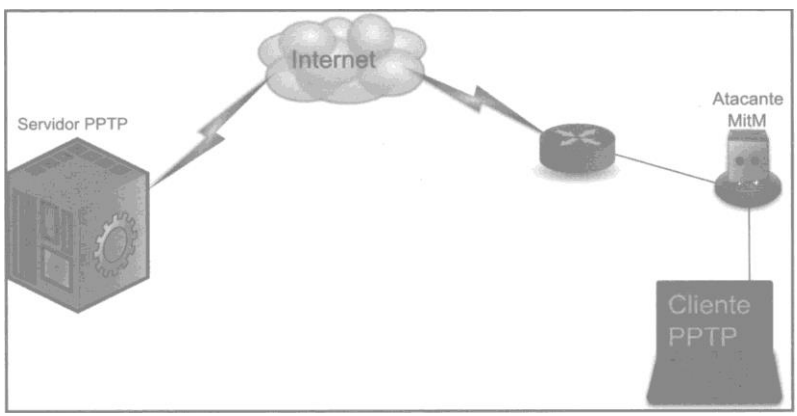


Fig 3.05: Escenario de un ataque MITM a una VPN de tipo PPTP.

El entorno para la prueba de concepto es el mostrado en la anterior imagen. En el que un usuario desea conectarse a un entorno corporativo, utilizando ***PPTP*** como mecanismo VPN, mientras se toma un café en un establecimiento público que ofrece WiFi gratuito.

Herramientas utilizadas

El atacante puede utilizar las dos herramientas detalladas a continuación para capturar los paquetes necesarios, utilizando como base el sistema operativo para auditorías de seguridad ***Backtrack.***

- ***Ettercap.*** Se trata de una herramienta destinada a la explotación de ataques ***MITM.*** En este caso será utilizada, junto con un ***plugin*** ya incorporado, para la captura de los datos de autenticación explotables en ***PPTP.*** Esta herramienta se puede utilizar además para ataques de ***DNS-spoofing*** entre otros.
- ***asleap.*** Es una herramienta que fue creada inicialmente para atacar el método de autenticación ***LEAP*** de Cisco. Posteriormente fue modificada para atacar el protocolo ***MSCHAPv2.*** Para el uso de ***asleap*** será necesario generar un fichero de hashes con el comando ***genkeys.***

Ataque MITM, Man in the Middle, para capturar la autenticación

Se inicia el proceso de ataque ***MITM*** con la herramienta ***Ettercap.*** Los pasos que hay que realizar para ello son los siguientes:

- Habilitar el ***sniffing*** en ***Ettercap*** para continuar con el resto del procedimiento.
- Seleccionar las víctimas del ataque. En este caso son la máquina del usuario que se conectará a la VPN y la puerta de enlace de la conexión pública.
- Iniciar el envenenamiento. Mediante la técnica ***ARP-Spoofing***, el atacante "se posiciona" entre las dos máquinas víctimas interceptando todas sus conexiones.
- Extraer los datos de autenticación mediante ***Ettercap.***

Una vez que el atacante configura debidamente ***Ettercap*** para interceptar la autenticación de PPTP aparecerá una pantalla como la siguiente en el momento que el cliente o víctima realiza dicha autenticación.

Fig 3.06: Configuración con *Ettercap* de un ataque MITM a una VPN PPTP.

El próximo paso el atacante consiste en realizar un ataque por fuerza bruta o diccionario al hash sobre los datos de autenticación obtenidos. En el caso de este ejemplo se realiza un ataque de ejemplo por diccionario. A continuación se muestra el resultado obtenido.

Una vez que se ha generado el fichero de hash y el de índices, se procede al ataque por diccionario del siguiente modo haciendo uso de ***asleap.***

```
# asleap -f hashes -n indexes -R C9:C5:CA:26:4C:A6:32:D4:64:B1:99:5E:AF:DE:BF:
10:7E:9F:99:D1:8F:C1:91:5F -C D8:7F:07:BA:A7:5C:A4:C8
```

Además de los ficheros mencionados anteriormente, se pasan como parámetros los datos del desafío y la respuesta obtenidos previamente con ***Ettercap.*** El resultado obtenido es el siguiente:

```
asleap 2.2 - actively recover LEAP/PPTP passwords. <jwright@hasborg.com>
        hash bytes:         619b
        NT hash:            35ccba9168b1d5ca6093b4b7d56c619b
        password:           testpass
```

En este ejemplo de ataque estaba todo demasiado preparado para que el ***password*** resultara sencillo de extraer. Aunque no significa que no sea relativamente sencillo en casos más reales, pues como se comentaba al inicio de esta sección se ha publicado un método que destroza toda la seguridad que pudiera ofrecer el protocolo ***MSCHAPv2.*** Eso sí, en el momento de escribir estas líneas se trata de un servicio de pago pues utiliza una potente infraestructura de cloud-computing con la que se descifran los datos interceptados en la autenticación de ***PPTP*** en cuestión de horas.

Como conclusión, se puede concluir que en el caso de utilizar ***PPTP*** debe hacerse utilizando un método aún más seguro como es el caso de certificados. Si por el contrario es posible modificar la tecnología de VPN de una infraestructura o bien implementar una nueva, es muy recomendable utilizar otras técnicas como pudieran ser ***L2TP*** sobre ***IPsec*** o el proyecto ***OpenVPN.*** En este caso se ha escogido la segunda opción por su relativa sencillez en la configuración, su flexibilidad y la seguridad que ofrece.

OpenVPN

OpenVPN es una solución de tecnología VPN equiparable a la tan extendida ***IPsec.*** Se trata de un proyecto open-source que utiliza como base las librerías ***OpenSSL*** para implementar sus mecanismos de cifrado y autenticación de usuarios o máquinas. A diferencia de ***IPsec***, ***Open VPN es*** una aplicación que se ejecuta en el nivel del usuario sin ser necesarias modificaciones a nivel de kemel y pila IP.

Entre las ventajas y funcionalidades que se pueden destacar están las siguientes:

- Solución robusta y segura. Se utilizan tecnologías más que testadas con lo que se asegura que los procesos de cifrado, autenticación e interacción con el sistema sean muy sólidos.

- Gran flexibilidad de configuración. Es posible utilizar Scripts y diversas configuraciones según el momento o estado de la conexión. Adicionalmente y entre otras muchas opciones, se puede configurar la llamativa opción de ofrecer el servicio VPN sobre el puerto 443/tcp como si se tratase de tráfico HTTPS. Esto último resulta muy útil para los clientes cuando se encuentran en entornos públicos y limitados solo a navegación web, así como entornos empresariales donde las configuraciones de los ***firewalls*** son muy estrictas.
- Diferentes métodos de autenticación. Es posible utilizar tanto autenticación por clave simétrica, como una infraestructura de clave pública que permitirá la utilización de certificados de usuario. Dichos certificados pueden ser auto-firmados o bien firmados por entidades certificadoras.
- Diferentes modos de trabajo. En este punto se definen dos conceptos de modo de trabajo diferentes. Por una parte el que atiende a la red y su configuración. Puede configurarse la VPN tunelizando desde la capa 2 en adelante o bien a partir de la capa 3. En el primer caso "se extiende" la red local a nivel de enlace, habiendo comunicación que broadcast, etc. Para este caso se utilizan los dispositivos virtuales ***TAP*** del kemel. En el caso de tunelizar la capa 3 que resulta el método más utilizado por su sencillez, se utilizan los dispositivos ***TUN*** y se trabaja a nivel de red.
- El otro modo de trabajo al se hace mención es la capacidad que posee ***OpenVPN*** para trabajar en modo ***P2P***, ***Peer-to-peer***, o bien en modo ***Server.*** El primero de ellos es el que se establece por defecto si no se indica lo contrario y resulta útil cuando se desean unir por ejemplo dos sedes o simplemente dos equipos bajo una conexión segura. El modo ***Server*** permite múltiples conexiones por parte de los clientes con tan solo un servidor ***Open VPN.***
- Sin problemas con ***NAT.*** Otro de los puntos fuertes que caracteriza a ***OpenVPN*** es la facilidad de trabajo frente a ***firewalls.*** A diferencia de otros servicios VPN, no existen problemas a la hora de realizar dobles ***NAT***\ agregar reglas, etc.

A día de hoy, la única contra que puede tener ***OpenVPN*** es la compatibilidad con dispositivos hardware comerciales. Aún no se ha dado el salto a incluir soporte ***OpenVPN*** nativo en muchos dispositivos, aunque es una tendencia que está variando y cada vez se está mejorando más en este aspecto.

Consideraciones previas a la configuración

Se omite la explicación de la instalación del servidor ***OpenVPN*** pues resulta de lo más sencilla. Por otra parte, el cliente oficial se encuentra en la mayoría de las ocasiones en los repositorios oficiales de cada distribución o bien en la web oficial de ***OpenVPN.*** En el caso de Debian bastaría la siguiente línea para instalar ***Open VPN.***

```
# apt-get install openvpn
```

Otro aspecto a tener en cuenta es la capacidad de poder utilizar dos modos de configuración y ejecución. ***OpenVPN*** puede ejecutarse tanto desde una terminal o bien como un demonio con

un script de inicio que lo gestione. En el caso del demonio se utilizan ficheros de configuración contenidos por defecto bajo el directorio ***/etc/openvpn.*** En el caso de la ejecución en una terminal, los valores de configuración se pasan como parámetros al comando ***OpenVPN.*** Para cualquier método utilizado, está disponible para el administrador un manual de ***Open VPN*** muy extenso y explicativo, con ejemplos de funcionamiento, así como ficheros de ejemplo situados en ***/usr/share/doc/openpvn/examples.*** Como habitualmente, para el manual se utiliza ***man openvpn.***

A continuación, ya que se ha introducido qué es ***OpenVPN***y algunas de sus características, se verán ejemplos de configuración. En todos los casos se utiliza el tunelizado a nivel de capa de red.

Configuración mediante clave pre-compartida

Para la utilización de la autenticación por clave pre-compartida se utilizará el modo ***P2P***, que es quizás el más sencillo que tiene ***Open VPN*** y el más rápido de configurar. Consiste en unir dos equipos remotos a través de una conexión segura. La contrapartida de este método resulta en que el servidor solo admitirá la conexión del equipo remoto, no podrá existir otro ***endpoint*** remoto conectado en paralelo utilizando el mismo socket o instancia de ***Open VPN*** en el mismo instante aunque este segundo posea las credenciales necesarias.

El entorno que se configurará será como el mostrado en la siguiente imagen.

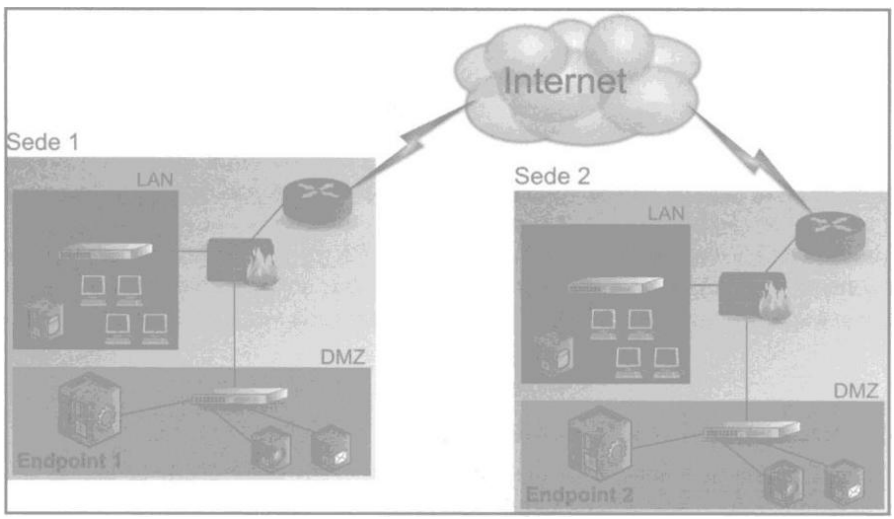


Fig 3.07: Esquema de una conexión VPN mediante modo P2P.

Es decir, se desean realizar las siguientes acciones:

- La máquina remota con IP 172.16.0.50 se conectará mediante VPN a la máquina local con IP 192.168.87.55.
- La IP interna de la VPN será 10.1.0.1 para la máquina local, el servidor, y 10.1.0.2 para el ***endpoint*** remoto.

- La máquina remota debería ser capaz de hacer ***ping*** a un host del exterior a través de la VPN, además de poder conectar con redes de la red 192.168.87.0/24 a la que se físicamente la máquina local, el servidor.

A continuación se muestra cómo debe configurarse ***OpenVPN*** para la infraestructura anterior. Los pasos seguidos, ficheros de configuración, etc. Por otra parte, será necesaria alguna configuración relacionada con ***iptables*** y ***enrutado.***

Por defecto, bajo el directorio de configuración ***/etc/openvpn*** no existe ningún fichero que establezca una configuración para ***OpenVPN.*** Es necesario crearlo manualmente y configurarlo de acuerdo a las necesidades de cada entorno.

El fichero ***/etc/openvpn/endpoint.conf*** quedaría del siguiente modo:

```
# Modo tun, nivel de red.
dev tun

# 10.1.0.1 es la máquina local
# 10.1.0.2 es el endpoint remoto
ifconfig 10.1.0.1 10.1.0.2

# La clave pre-compartida
secret preshared.key
```

Atendiendo a la línea que comienza con el parámetro ***secreta*** es necesario generar ese fichero de clave estática. Para ello se utiliza el comendo ***openvpn*** del siguiente modo.

```
# openvpn --genkey --secret /etc/openvpn/preshared.key
```

Con lo anterior se genera el fichero ***/etc/openvpn/preshared.key*** que contiene una clave estática de 2048 bits. Esta clave debe ser compartida de forma segura con el ***endpoint*** remoto utilizando por ejemplo ***scp*** como medio de transporte o un ***pendrive*** cifrado en caso de poder instalarse físicamente en el equipo remoto. Es necesario comprender que ese fichero no puede caer en manos de terceros pues dispondrá de la capacidad de conexión con el equipo y el descifrado de información cifrada que haya podido ser capturada. Deben regularizarse los permisos del fichero de clave para que queden como 600, puede hacerse del siguiente modo.

```
# chmod 600 /etc/openvpn/preshared.key
```

En el lado del ***endpoint*** remoto, la el fichero de configuración de ***openvpn*** debería ser como el siguiente:

```
# Modo tun, nivel de red.
dev tun

# 10.1.0.2 es la máquina local
# 10.1.0.1 es el endpoint remoto
ifconfig 10.1.0.2 10.1.0.1

# La clave pre-compartida
secret preshared.key

# La dirección remota del endpoint al que se debe conectar OpenVPN
remote 192.168.87.55

# Se establece como puerta de enlace por defecto el endpoint remoto a través de
# del dispositivo tunX
redirect-gateway def1
```

Una vez que en ambas máquinas se reinicien los demonios ***Open VPN*** con los ficheros de configuración anteriormente establecidos, deberían obtenerse los siguientes resultados al lanzar el comando ***ifconfig.*** Primero se muestra el caso de la máquina que actúa como "servidor" y posteriormente la máquina "cliente".

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-
00-00
          inet addr:10.1.0.1  P-t-P:10.1.0.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1142 (1.1 KiB)  TX bytes:0 (0.0 B)
```

En el endpoint remoto se obtiene el siguiente resultado.

```
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-
00-00
          inet addr:10.1.0.2  P-t-P:10.1.0.1  Mask:255.255.255.255
```

```
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 B)  TX bytes:1142 (1.1 KiB)
```

El mismo ejemplo de configuración de ***endpoint*** remoto desde ***Gnome 3*** tendría el siguiente aspecto.

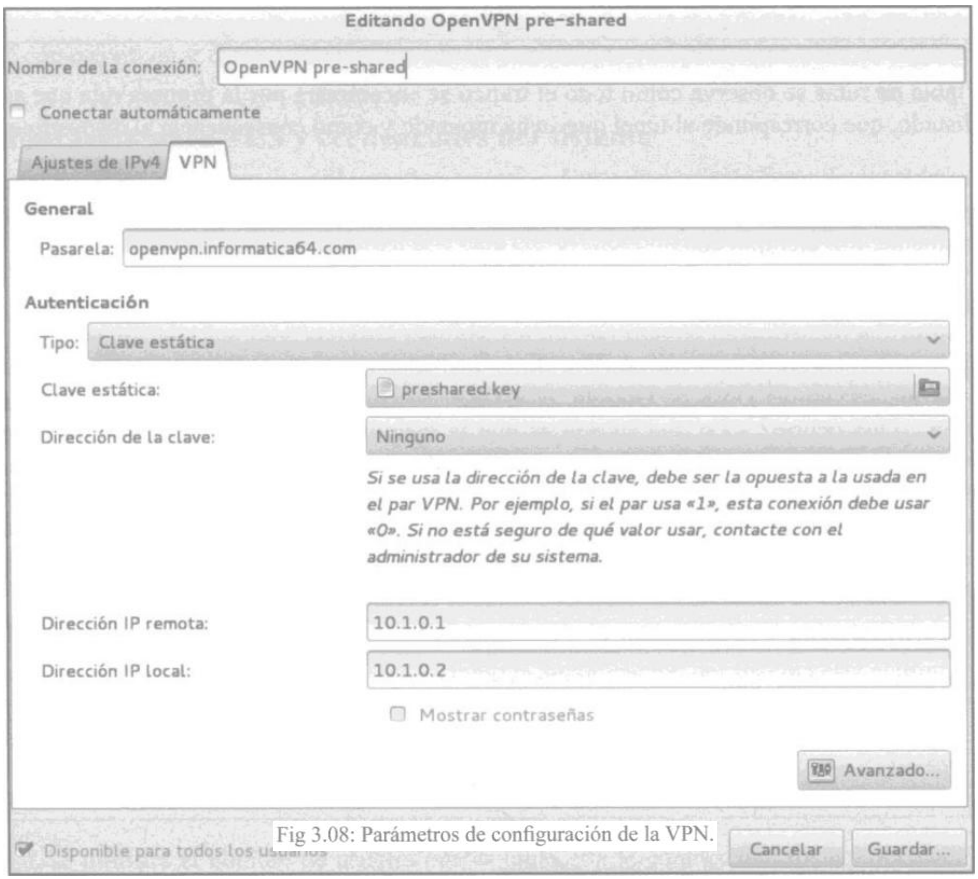


Fig 3.08: Parámetros de configuración de la VPN.

En la salida de los comandos ifconfig ejecutados en ambas máquina se comprueba que se ha creado el túnel de forma correcta y como consecuencia los dispositivos ***tun***, en el caso del ejemplo ***tunO*** pues son los únicos existentes. Antes de comprobar que se tiene conectividad con la máquina remota desde el cliente se ejecuta el comando ***route -n*** para verificar que la tabla de enrutamiento de la máquina "cliente" es correcta.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.1.0.1        128.0.0.0       UG    0      0        0 tun0
0.0.0.0         172.16.0.1      0.0.0.0         UG    0      0        0 eth0
10.1.0.1        0.0.0.0         255.255.255.255 UH    0      0        0 tun0
128.0.0.0       10.1.0.1        128.0.0.0       UG    0      0        0 tun0
172.16.0.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.87.55   172.16.0.1      255.255.255.255 UGH   0      0        0 eth0
```

En la tabla de rutas se observa cómo todo el tráfico se encaminará por la primera ruta que aparece en el listado, que corresponde al túnel que se ha montado y como consecuencia al dispositivo ***tunO.***

Tras establecer el túnel, la máquina que actúa como servidor no está capacitada para enrutar paquetes que provengan de otra. Es necesario convertir la máquina en router tal y cómo se explicó anteriormente. Por ejemplo con:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Ahora que todo está correctamente establecido, se realiza una prueba de***ping*** al exterior. En concreto a uno de los servidores DNS de Google, el 8.8.8.8. Pero parece que no funciona, no se obtiene respuesta. ¿Qué ocurre? - Lo que sucede es que la máquina que actúa como servidor envía los paquetes que provienen de la máquina cliente, a través del túnel, hacia su puerta de enlace. Los paquetes llegan correctamente a la dirección 8.8.8.8; pero cuando la puerta de enlace que proporciona acceso a internet, la establecida por defecto en la máquina servidor, no sabe a dónde tiene que enviar los paquetes de la red de la VPN. La tabla de rutas de la máquina servidor tiene un aspecto como el siguiente:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.0.2        0.0.0.0         255.255.255.255 UH    0      0        0 tun0
192.168.87.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
0.0.0.0         192.168.87.1    0.0.0.0         UG    0      0        0 eth0
```

Existen dos soluciones para el problema anterior.

- Establecer una ruta estática en la puerta de enlace. Se establecería que el tráfico destinado a la subred 10.1.0.0/16 iría destinado a la IP 192.168.87.55, que no es más que la máquina que actúa como servidor ***OpenVPN.***
- Realizar SNAT en la máquina que actúa como servidor ***OpenVPN*** para los paquetes que provengan de la IP 10.1.0.2, la del cliente.

Se opta por la segunda opción pues resulta más sencilla y rápida de implementar. Lo único que habría que hacer en el servidor ***Open VPN*** consistiría en ejecutar la siguiente orden.

```
# iptables -t nat -A POSTROUTING -o eth0 -s 10.1.0.2 -j MASQUERADE
```

En este punto ya estaría todo configurado para hacer funcionar el entorno planteado inicialmente. Como se puede observar, no resulta una tarea compleja configurar ***OpenVPN*** para utilizar "autenticación" y cifrado mediante el uso de clave pre-compartida. Resulta un buen método cuando se quieren conectar dos máquinas remotas entre sí, pero en el caso de disponer de más clientes tipo ***road-warriors***, etc. no es posible utilizar una configuración de este tipo. En los próximos ejemplos se verá cómo configurar ***OpenVPN en*** modo server y utilizando ***TLS*** para establecer un canal seguro.

Configuración para TLS y credenciales del sistema

La configuración anterior resulta útil cuando se desea enlazar dos sedes o dos máquinas utilizando una conexión segura. El principal problema que posee es la falta de flexibilidad y seguridad en cuanto a la autenticación. Elay que recordar que para que el ***endpoint*** remoto estableciese la conexión se debía compartir previamente la clave pre-compartida. Eso ya supone un problema si no se realiza a través de un medio seguro. Por otra parte, si dicha clave se extravía o cae en manos ajenas o de un atacante, podrá acceder a los recursos remotos sin ningún tipo de dificultad. Y por último y no menos importante, existe el problema de que se trata de una conexión p2p por lo que no es posible establecer un entorno ***road-warrior*** como se desea.

A continuación se expone la creación de un entorno con servidor ***OpenVPN*** en el que podrán conectarse múltiples clientes de forma simultánea. La idea queda plasmada en la figura inicial de este capítulo, donde se mostraba una infraestructura ***road-warrior.***

La autenticación del usuario se realizará mediante el mecanismo de autenticación que tenga configurado el sistema operativo. Esto puede parecer muy complejo pero todo queda simplificado cuando en el sistema operativo donde se ejecuta el servidor ***OpenVPN*** se utiliza ***PAM, Pluggable Authentication Modules.*** En el caso de este ejemplo la autenticación del sistema está configurada de forma clásica y se utiliza el fichero ***/etc/passwd.*** Una de sus líneas contiene la siguiente información, que será la usada a la hora de la prueba de autenticación.

```
carlos:x:1001:1001:,,,:/home/carlos:/bin/bash
```

Es decir, se utilizará el nombre de usuario ***carlos*** junto con la contraseña asociada en el fichero / ***etc/shadow*** para iniciar sesión en el servidor ***Open VPN.***

A priori puede parecer que al utilizar este método no se cifrará la conexión o no se tendrá la misma seguridad que en el caso anterior, pues el cliente tan solo precisa de un usuario y una contraseña. En este momento es donde entra en escena uno de los mecanismos implementados en ***OpenVPN*** para el inicio de sesión, establecimiento del canal seguro, etc. Se trata del protocolo criptográfico Diffie-

Hellman que, a modo de resumen, consiste en el establecimiento de claves en un medio inseguro y entre partes que no han mantenido un contacto previo.

Creación de certificados y Diffie-Hellman

Para el configurar ***Open VPN*** en modo servidor es necesario generar un certificado de servidor además de los parámetros para el mecanismo ***Diffie-Hellman.*** El certificado puede ser auto-firmado con una ***CA*** de creación propia, que es lo más sencillo. Tanto para crear una nueva ***CA*** y los certificados se puede utilizar por su sencillez la herramienta ***easy-rsa*** que está incluida en la instalación de ***OpenVPN.*** El procedimiento queda resumido en los siguientes puntos:

- Configuración de las variables para el uso de ***easy-rsa.***
- Creación de la ***CA.***
- Creación del certificado del servidor emitido por la ***CA*** creada.
- Creación de los parámetros para ***Diffie-Hellman.***

El directorio de trabajo para ***easy-rsa*** es ***/usr/share/doc/openvpn/examples/easy-rsa/2.0.*** La primera tarea a realizar consiste en modificar el fichero ***vars.*** Tan solo es necesaria la modificación de los siguientes parámetros.

```
export KEY_COUNTRY="ES"
export KEY_PROVINCE="MA"
export KEY_CITY="Mostoles"
export KEY_ORG="Informatica64"
export KEY_EMAIL="calvarez@informatica64.com"
```

El primero paso a realizar tras la modificación de parámetros consiste en establecer dichas variables ejecutando el script ***vars.***

```
# . /vars
```

A continuación se adjunta la información visualizada en la pantalla a la hora de crear tanto la ***CA*** como el certificado del servidor.

```
root@squeetest:/usr/share/doc/openvpn/examples/easy-rsa/2.0# ./build-ca
Generating a 1024 bit RSA private key
.....................++++++
.....................................................++++++
writing new private key to 'ca.key'
-----
 ...
-----
```

```
Country Name (2 letter code) [ES]:
State or Province Name (full name) [MA]:
Locality Name (eg, city) [Mostoles]:
Organization Name (eg, company) [Informatica64]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [Informatica64 CA]:
Name []:
Email Address [calvarez@informatica64.com]:

root@squeetest:/usr/share/doc/openvpn/examples/easy-rsa/2.0#       ./build-key-
server i64ovpn
Generating a 1024 bit RSA private key
............++++++
........++++++
writing new private key to 'i64ovpn.key'



Country Name (2 letter code) [ES]:
State or Province Name (full name) [MA]:
Locality Name (eg, city) [Mostoles]:
Organization Name (eg, company) [Informatica64]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [i64ovpn]:
Name []:
Email Address [calvarez@informatica64.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /usr/share/doc/openvpn/examples/easy-rsa/2.0/openssl.
cnf
Check that the request matches the signature
```

```
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :PRINTABLE:'MA'
localityName          :PRINTABLE:'Mostoles'
organizationName      :PRINTABLE:'Informatica64'
commonName            :PRINTABLE:'i64ovpn'
emailAddress          :IA5STRING:'calvarez@informatica64.com'
Certificate is to be certified until Dec 25 19:28:02 2022 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Se ha creado la ***CA*** con los datos establecidos en las variables. Posteriormente, utilizando esa ***CA*** se ha generado un certificado de servidor llamado i64ovpn, utilizando de igual modo los parámetros previamente establecidos.

Por último, antes de configurar el servidor ***Open VPN,*** es necesaria la generación de los parámetros para ***Diffie-Hellman.*** Es un proceso bastante más lento pues entre otras operaciones se calcula un número primo muy grande. Para realizar dicha operación se utiliza la orden:

```
root@squeetest:/usr/share/doc/openvpn/examples/easy-rsa/2.0# ./build-dh
```

Los ficheros generados se encuentran bajo el directorio ***/usr/share/doc/openvpn/examples/easy-rsa/2.0/keys.*** En el fichero de configuración se visualizarán cuáles son los necesarios, que se copiarán en el directorio de nueva creación ***/etc/openvpn/keys.***

Configuración del servidor

En el lado del servidor se establecerá la configuración en cualquier fichero terminado en .ccw/bajo el directorio ***/etc/openvpn.*** En este caso se crea el fichero ***/etc/openvpn/i64.conf*** con el siguiente contenido.

```
# Modo tun
dev tun

# CA, Certificado y Diffie-Hellman
ca keys/ca.crt
```

```
cert keys/i64ovpn.crt
key keys/i64ovpn.key
dh keys/dh1024.pem

# Configuración como servidor utilzando la subred 10.1.0.0/24
server 10.1.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt

# No es necesario el certificado del cliente
client-cert-not-required

# Autenticación usando PAM
plugin /usr/lib/openvpn/openvpn-auth-pam.so common-auth
```

Salvo la línea en la que se especifica que se está creando una ***VPN*** con dispositivo ***TUN,*** el resto se ha modificado por completo. Las líneas que corresponden a la definición de los ficheros recién creados están claras con los comentarios del fichero de configuración. Por otro lado, la directiva server es la que modifica el comportamiento de ***Open VPN*** para que se comporte en modo servidor, indicando la subred en la que trabajará ***Open VPN.*** El servidor siempre se asignará la IP terminada en .1 de la subred definida.

Lo especial de esta configuración es que se requiere que los clientes utilicen certificado y por lo tanto utilicen la autenticación por ***PAM*** definida en el sistema. Dicho comportamiento corresponde a la dos últimas directivas del fichero de configuración.

Configuración del cliente

Se genera el fichero de configuración ***/etc/openvpn/client.conf*** y se incluye el siguiente contenido en el mismo.

```
## Configuración del cliente
# Dispositivo tun
dev tun

# Modo cliente y servidor remoto
client
remote openvpn.informatica64.com

# CA para verificar el servidor
```

```
ca keys/ca.crt

# Autenticación por password
auth-user-pass

# Salida hacia Internet por la VPN
redirect-gateway def1
```

El cliente debe disponer del fichero ***crt*** de la ***CA*** para poder comprobar la autenticidad del servidor al que se está conectando. Una vez que se cargue la configuración en la máquina cliente se establecerá el túnel y se modificará la tabla de rutas de acuerdo a la configuración establecida.

```
root@wizi:~# service openvpn start
[....] Starting virtual private network daemon: clientEnter Auth Username:carlos
Enter Auth Password:
. ok
root@wizi:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.1.0.5        128.0.0.0       UG    0      0        0 tun0
0.0.0.0         172.16.0.1      0.0.0.0         UG    0      0        0 eth0
10.1.0.1        10.1.0.5        255.255.255.255 UGH   0      0        0 tun0
10.1.0.5        0.0.0.0         255.255.255.255 UH    0      0        0 tun0
128.0.0.0       10.1.0.5        128.0.0.0       UG    0      0        0 tun0
172.16.0.0      0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.87.55   172.16.0.1      255.255.255.255 UGH   0      0        0 eth0
root@wizi:~#
```

Fig 3.09: Tabla de enrutamiento.

En la pantalla anterior se verifica que se ha solicitado un usuario y contraseña para poder establecer la conexión al servidor.

Configuración TLS. Autenticación mediante certificados

Con la configuración anterior se conseguía autenticación mediante usuario y contraseña, además de la verificación de autenticidad del servidor. Para dotar de más seguridad y flexibilidad a un servidor ***OpenVPN***, debe usarse el método detallado a continuación. Por cada usuario se creará un certificado que le proporcionará acceso al servidor.

Generación de los certificados de los clientes

El primer paso a realizar para hacer funcional esta configuración consiste en generar los certificados de los clientes que vayan a utilizar la ***VPN.*** Para ello se utilizará de nuevo las herramientas proporcionadas por ***easy-rsa.*** La configuración de las variables, la ***CA***, el certificado del servidor y los parámetros ***DH***, serán los mismos que en el ejemplo anterior, por lo que solo es necesaria la generación de los certificados de los clientes.

Es posible generar los certificados de los clientes de dos modos. Pueden estar protegidos por una contraseña o frase de paso para su desbloqueo, o bien sin protección alguna. Se escoge el segundo de los modos para ejemplo.

Se vuelve a repetir el proceso de la carga de las variables utilizadas por easy-rsa antes de crear el certificado de un usuario. Para ello es necesario situarse en el directorio ***/usr/share/doc/openvpn/examples/easy-rsa/2.0/*** y ejecutar el script ***vars.*** Acto seguido se utiliza el script ***build-key-pass***. La creación del certificado para el usuario ***car los*** tendría una salida como la siguiente.

```
root@squeetest:/usr/share/doc/openvpn/examples/easy-rsa/2.0# ./build-key carlos
Generating a 1024 bit RSA private key
..++++++
.....++++++
writing new private key to 'carlos.key'
-----
 ...
-----
Country Name (2 letter code) [ES]:
State or Province Name (full name) [MA]:
Locality Name (eg, city) [Mostoles]:
Organization Name (eg, company) [Informatica64]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) [carlos]:
Name []:Carlos Alvarez Martin
Email Address [calvarez@informatica64.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /usr/share/doc/openvpn/examples/easy-rsa/2.0/openssl.
cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'ES'
stateOrProvinceName   :PRINTABLE:'MA'
```

```
localityName            :PRINTABLE:'Mostoles'
organizationName        :PRINTABLE:'Informatica64'
commonName              :PRINTABLE:'carlos'
name                    :PRINTABLE:'Carlos Alvarez Martin'
emailAddress            :IA5STRING:'calvarez@informatica64.com'
Certificate is to be certified until Dec 27 17:04:07 2022 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Comprobando el contenido del directorio keys dentro se verifica que existe el certificado para ***carlos*** cuyos ficheros son ***carlos.key*** y ***carlos.crt.*** El primero de ellos es la clave privada que debe ser protegida convenientemente.

Fichero de configuración del servidor

El nuevo fichero de configuración ***/etc/openvpn/i64.conf*** incluiría las siguientes directivas.

```
# Dispositivo tun.
dev tun

# Certificados y Diffie-Hellman
ca keys/ca.crt
cert keys/i64ovpn.crt
key keys/i64ovpn.key
dh keys/dh1024.pem

# El servidor actúa como servidor para el establecimiento del canal seguro
tls-server

# Configuración como servidor utilzando
# la subred 10.1.0.0/24
server 10.1.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
```

Como puede apreciarse la configuración básica es muy sencilla. Una vez guardada la información anterior y reiniciado el servicio ***OpenVPN*** tan solo quedaría configurara el cliente del siguiente modo.

Fichero de configuración del cliente

En el paso de generación de certificados se generó un certificado para el usuario ***cavíos***. Los ficheros correspondientes a dicho certificado, junto con el certificado de la CA, deben estar disponibles en la máquina cliente. Como ocurría en el caso de la clave pre-compartida, es altamente recomendable utilizar un medio seguro para la adquisición de los certificados por parte del cliente.

Una vez obtenidos los ficheros necesarios, el cliente debe generar un fichero de configuración como el siguiente. En este ejemplo se trata del fichero ***/etc/openvpn/carlos.conf.***

```
## Configuración del cliente
# Dispositivo tun
dev tun

# Modo cliente y servidor remoto
client
remote openvpn.informatica64.com

# CA para verificar el servidor
ca keys/ca.crt

# Datos de usuario
cert keys/carlos.crt
key keys/carlos.key

# Salida hacia Internet por la VPN
redirect-gateway def1
```

En el momento en que se cargue la configuración anterior, ***OpenVPN*** establecerá el túnel privado habiendo hecho uso de certificados tanto para el servidor como en el cliente. Con ello se ha conseguido privacidad pues se está cifrando el canal de comunicaciones en el que se negociará una clave simétrica; y autenticación por el hecho de utilizar un certificado de cliente.

Algunas directivas para fortificar OpenVPN

La última configuración realizada se puede considerar bastante robusta, por supuesto a menos que los certificados del cliente caigan en manos de un atacante. A pesar de ello, es posible utilizar una

serie de parámetros de configuración que permitirán crear un entorno ***OpenVPN*** más robusto. A continuación se detallan algunos de ellos.

Evitar ataques DoS

Es posible mitigar ataques ***DoS*** contra el servidor ***Open VPN***utilizando un par de parámetros internos. La finalidad de la utilización de los mismos consistirá en aceptar solo conexiones lícitas y tratar de descartar las que sean ataques, escaneos, etc. Por supuesto todo esto se puede combinar con el uso de ***iptables*** u otras herramientas, pero aquí se muestra qué hay configurar en ***OpenVPN.***

- ***tls-auth <fichero> [dirección].*** Consiste en utilizar una clave simétrica para poder comenzar con la negociación ***TLS.*** Es decir, utilizando este método, solo aquellos clientes que dispongan de la clave pre-compartida podrá hacer el intento de establecimiento de sesión ***TLS*** con los certificados de usuario. Puede utilizarse la clave pre-compartida del primer ejemplo para este parámetro de configuración.
- ***connect-freq <n_conexiones> <n_segundos>.*** Se trata de limitar el número de conexiones que podrá atender ***OpenVPN.*** Se indican el número máximo de conexiones por cada ***n*** segundos. Este parámetro es solo utilizable cuando ***OpenVPN está*** usando ***UDP*** como capa de transporte. En el caso de ***TCP***, cuando es necesario evitar***firewalls*** por ejemplo, debe usarse la directiva ***max-clients.*** En ambos cabe la posibilidad que clientes lícitos no puedan establecer conexiones por la limitación establecida. Sin lugar a dudas es más recomendable utilizar ***tls-auth.***

Evitar ataques MiTM

Una de las principales amenazas cuando se realizan conexiones en entornos públicos u hostiles son los ataques de ***hombre-de-en-medio.*** Si el ataque resulta exitoso, el atacante podría suplantar la identidad de un servidor ***Open VPN*** y hacer creer a la víctima que el servidor al que se conecta es lícito. Existen una serie de parámetros que ayudarán a evitar o detectar este tipo de ataques.

- ***ns-cert-type server/client.*** Ayudará a detectar si el extremo al que se realiza la conexión es un servidor. Se utiliza la marca ***nsCertType*** del certificado para verificar si es de tipo ***server*** o ***client.*** En el cliente se establecería el valor a server para verificar el certificado del servidor.
- ***tls-verijy script/comando*** y ***tls-remote nombre.*** En ambos casos se comprueban los certificados del servidor y el cliente. En el primero de ellos se puede utilizar un script que interactúe con ***OpenVPN*** mediante variables de entorno y poder comprobar si el certificado es válido, la ***CA*** que lo emitió, etc. En el segundo caso, es posible especificar un ***CN, Common Ñame***, para aceptar solo conexiones de esa máquina especificadas por el certificado que presenta la misma al iniciar la conexión.

Existen muchísimas configuraciones posibles para ***OpenVPN***, combinaciones de directivas, métodos de autenticación alternativos, etc. En este punto se han tratado de mostrar las configuraciones básicas en cuanto a autenticación, ***routing***, etc. Si está interesado en saber mucho más acerca de ***OpenVPN*** y sus diferentes conexiones es posible utilizar su ***man,*** que es de los más completos que hay entre muchas aplicaciones.

3.Monitorización de la red

Otro de los aspectos a tener en cuenta a la hora de securizar una infraestructura de red es conocer en todo momento cuál es el estado de sus componentes. Desde la electrónica del entorno hasta aplicaciones, servicios, etc. Muchos de los ataques que se realizan a las máquinas elevan el consumo de recursos de una máquina y de la red, por lo que es fundamental disponer de una visión del estado de la red para detectar anomalías en el funcionamiento del entorno. Existen diferentes protocolos para monitorizar cada uno de los componentes. Un claro ejemplo es el caso de ***SNMP***, ***Simple NetWork Management Protocol***, un estándar en cuanto a la monitorización de dispositivos de red muy extendido. La mayoría de los dispositivos comerciales, electrónica de red, etc.; que pueden ser gestionados, implementan ***SNMP*** para la gestión del dispositivo.

Independientemente del protocolo y/o software utilizado para implementar un sistema de monitorización de la red, se destacan los siguientes componentes fundamentales.

- Elemento de red (***NetWork element***, ***NE)*** o dispositivo administrado. Se trata de los elementos sobre los que se desea obtener determinada información. Es posible que a veces solo sea necesario averiguar si el dispositivo responde y conectado a la red, o bien obtener más detalles sobre su funcionamiento. En el primero de los casos bastaría con una simple comprobación mediante peticiones ***ping*** (***echo requests***), sin embargo para obtener más información sería necesaria la disponibilidad de agentes en cada elemento de red, ***NE.*** Como se comentaba anteriormente, los elementos de la red pueden ser servidores, impresoras, switches gestionables, etc.
- Agente. Es un componente software integrado y utilizado en los dispositivos gestionados. Su uso proporciona acceso a recursos del sistema, errores, datos de red, salud, etc.
- Sistema administrador de red (***NetWork Management System***, ***NMS).*** Se refiere al sistema o conjunto de sistemas en los que se ejecuta el software encargado de la monitorización de la red.

Existen diversas herramientas con las que monitorizar un entorno de red y propósitos diferentes. Por ejemplo, en el caso de desear monitorizar el uso de la red se puede utilizar tanto el tradicional ***MRTG,*** ***Multi Router Traffic Grapher*** o bien un software más reciente como ***CACTI.*** En el caso de monitorización de sistemas, incluyendo servicios, salud, actualizaciones, etc. se hará uso de ***Icinga,*** aunque existen más proyectos destinados al mismo cometido como pueden ser ***Nagios*** o ***PandoraFMS.*** Para todos los casos es posible e incluso necesaria la utilización del ya mencionado protocolo ***SNMP***, por lo que siempre es interesante disponer de dispositivos gestionables que implementen dicho protocolo.

Icinga

Icinga es un software de monitorización ***Open Source***, ampliamente utilizado en las infraestructuras de red para velar por su monitorización y buen funcionamiento. Se trata de un ***fork*** del conocido

Nagios. Es por ello que comparte su misma estructura, ***plugins*** y su funcionamiento es prácticamente el mismo.

Se sigue el mismo esquema base que en la explicación general anterior. Es decir, ***Icinga*** es el ***NMS*** o sistema administrador de red que verifica el estado ***NE*** o elementos de red haciendo uso o no de unos agentes. En el caso de ***Icinga,*** al igual que ocurre con ***Nagios***, no se utiliza ***SNMP*** por defecto para la obtención de datos en las máquinas monitorizadas. En este caso se utiliza ***NRPE***, ***Nagios Remóte Plugin Executor***, que permitirá a ***Icinga*** la obtención de información cuando esté programada.

Icinga es una herramienta de monitorización multipropósito, con la que se podrá verificar desde la conectividad con una máquina o dispositivo, hasta conocer las actualizaciones disponibles existentes en un entorno ***Debían***. Todo ello se puede realizar mediante funcionalidades incluidas en el ***core*** de la aplicación o bien mediante el uso de ***plugins.*** Como se comentaba anteriormente, dichos ***plugins*** guardan compatibilidad con ***Nagios.***

Instalación de Icinga

Como ocurre en todas las distribuciones y con todas las aplicaciones ***Open Source***, es posible instalar ***Icinga*** de dos modos. Ya sea compilando la aplicación o instalándola desde los repositorios adecuados para la distribución utilizada, oficiales o no. A pesar de que se trata de una aplicación de fácil instalación, se ha querido hacer hincapié en este punto para detallar qué componentes se instalan por dependencias con ***Icinga.***

```
# apt-get install icinga
```

Se instalará el software deseado más algunas dependencias entre las que se destacan las siguientes, que serán usadas por ***Icinga*** para su funcionamiento:

- ***apache2.*** Se utiliza ***Apache*** como servidor web por defecto para acceder a la interfaz de ***Icinga.***
- ***Plugins.*** Se instalan ***plugins*** iniciales para realizar comprobaciones de servicios utilizando ***tcp, smtp***, ***pop3***, ***http***, ***ping***, etc. Entre otras funciones importantes de los ***plugins*** básicos instalados se encuentran la gestión de notificaciones de eventos vía e-mail, además de la gestión de disparadores para solventar problemas de forma proactiva.
- ***Samba.*** Para la integración con redes ***Windows***, se instalan dependencias y el cliente para ***Samba.*** De hecho, durante la instalación de ***Icinga*** se solicitará la introducción de un grupo de trabajo si no hubiera uno ya configurado.

Para acceder a la consola de administración de ***Icinga*** se utiliza la dirección ***http://<host>/icinga*** en el que se solicitará un ***login*** básico de Apache. Los datos de dicho ***login*** se introdujeron durante la instalación de ***Icinga*** y pueden ser modificados a posteriori mediante la aplicación ***htpasswd*** proporcionada por el paquete ***Apache.*** Una de las vistas de la interfaz podría ser la siguiente.

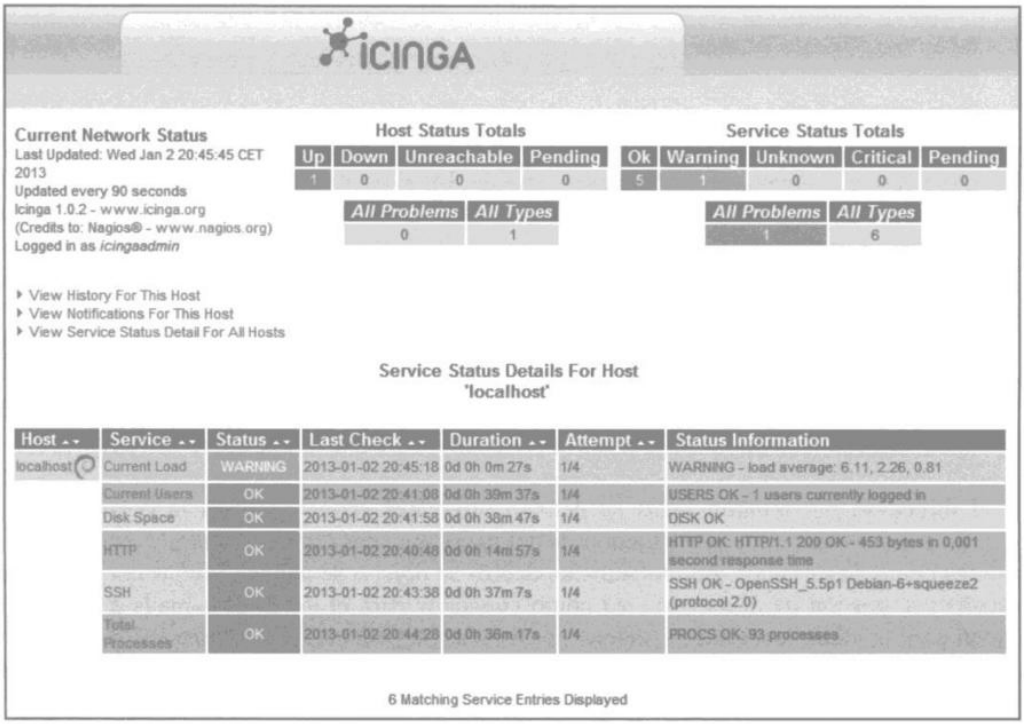


Fig 3.10: Estado de la máquina local. Configuración por defecto.

Configuración básica de Icinga. Monitorizar una máquina remota

Con solo instalar ***Icinga*** es posible monitorizar el estado de la máquina local que lo ejecuta, así como los servicios más importantes instalados en la misma. Pero la potencia de la monitorización de un entorno de red viene dada precisamente por la gestión y visualización de todos sus elementos remotos. Aunque antes de pasar a configurar ***Icinga,*** es necesario mencionar los ficheros de configuración más importantes y cuál es su cometido en el funcionamiento de ***Icinga.***

Ficheros de configuración

El directorio de configuración por defecto es /***etc/icinga***, aunque puede ser modificado para utilizar otro en el fichero ***/etc/default/icinga*** en el caso de ***Debían.*** Los ficheros más relevantes que se encuentran en el directorio de configuración son los siguientes.

- ***/etc/incinga/incinga.cfg.*** Es el fichero de configuración general de ***Icinga.*** En él se definen diversos aspectos como pueden ser el usuario y grupo que ejecuta ***Icinga*** en el sistema, configuraciones para la integración con ***rsyslog***, parámetros generales de la interfaz web y un largo etcétera. También es posible definir los directorios para ficheros temporales, de configuración de los objetos de ***Icinga***, etc.

- ***/etc/icinga/cgi.cfg.*** Fichero destinado a la configuración del comportamiento de los ***CGI*** de ***Icinga.*** Uno de los aspectos más destacables de este fichero es la gestión de los roles para usuarios con diferentes privilegios. También se configuran determinados comportamientos de la interfaz web.
- ***/etc/icinga/apache2.conf*** y ***/etc/icinga/htpasswd.*** En ambos ficheros se configura la parte correspondiente al servidor web ***Apache.*** En el primero de ellos se definen directorios de trabajo de apache, ejecución de ***CGIs***, ficheros de log, etc. En el segundo es donde se almacenan los usuarios definidos con ***htpasswd.*** Obviamente para esto último es necesario indicar al binario ***htpasswd***, la ruta del fichero ***/etc/icinga/htpasswd***, ya que no se trata de una aplicación exclusiva de ***Icinga.***
- ***/etc/icinga/commands.cfg.*** En este fichero se definen comandos personalizados para usarlos posteriormente en diversas situaciones. Uno de los comandos definidos por defecto es el que se encarga de la notificación por correo electrónico. En dicho ejemplo se realiza una redirección de un mensaje hacia el comando ***mailx*** que se encargará de enviar el correo.

Bajo el directorio ***/etc/icinga/objects*** se definen diferentes elementos para la configuración de ***Icinga.*** Entre esos elementos se encuentran la definición de servidores, servicios y grupos tanto de servidores como de servicios. Adicionalmente dentro de ***ohjects*** se definen franjas horarias a modo de estadística, gestión de ***SLAs***, etc., así como contactos para el envío de e-mails e información adicional para la identificación de sistemas operativos. Comentado todo lo anterior, son relevantes para la configuración de ***Icinga*** los siguientes ficheros que pueden estar estructurados de forma diferente, dependiendo de la distribución utilizada.

- ***contacts icinga. cfg.*** Se definen los contactos y la forma de enviar notificaciones a los mismos. Es posible definir grupos de administradores, periodos de notificaciones, etc. Serán usados en las definiciones de los hosts y/o servicios.
- ***generic-host icinga. cfg*** y ***generic-serviceicinga.cfg.*** Son ficheros utilizados como plantilla para la definición de servicios y máquinas a posteriori. En dichas bases se definen valores como el periodo de notificación, notificaciones, avisos de ***flap*** o fluctuación en un host o servicio, grupos de contacto, disparadores, etc.
- ***hostgroups icinga.cfg.*** Es un fichero utilizado para la definición de grupos de servidores. Por defecto se definen grupos de servidores ***http***, ***ssh*** y ***debían.*** Posteriormente podrán utilizarse para definir servicios y su correspondiente monitorización para los grupos de servidores indicados.
- ***localhosticinga. cfg.*** Es el fichero en el que se define el host ***localhost***, así como los servicios que se monitorizan en dicho host. Es decir, en ese fichero se está definiendo la máquina que ejecuta ***Icinga.*** Es posible que la máquina local posea servicios comunes con otras máquinas como puede ser un servidor ***ssh*** o bien un servidor web. Para esto último se definen los servicios para los ***hostgroups*** definidos. En el caso de esta instalación en concreto se trata del fichero ***Services Jcinga. cfg.***
- ***Services Jnciga.cfg.*** Se definen servicios comunes para los grupos de servidores indicados en el fichero ***hostgroups icinga. cfg.***

A la hora de definir un nuevo host, como es la finalidad de este ejemplo, pueden utilizarse como base los ficheros de configuración originales y personalizarlos a las necesidades oportunas.

Definiendo un nuevo host y sus servicios monitorizados

Para definir un host de prueba llamado *penguin*, se crea un nuevo fichero llamado */etc/icinga/objects/penguin_icinga.cfg* partiendo de la base de *localhost_icinga.cfg*. El contenido del fichero quedaría del siguiente modo:

```
# Definición del servidor - penguin -
# ----------------------------------
define host{
        use                     generic-host            ; Configuración base
        host_name               penguin
        alias                   penguin
        address                 penguin.informatica64.com
        }

# Alerta de especio en disco.
# -- Advertencia entre 20% y 10% libre
# -- Crítico por debajo de un 10%
define service{
        use                             generic-service    ; Configuración base
        host_name                       penguin
        service_description             Espacio en disco
        check_command                   check_nrpe!check_all_disks!20%!10%
        }

# Verificación de la carga del servidor
define service{
        use                         generic-service       ; Configuración base
        host_name                   penguin
        service_description         Carga actual
                check_command

check_nrpe!check_load!5.0!4.0!3.0!10.0!6.0!4.0
        }
```

```
# Verificación de actualizaciones
define service{
        use                             generic-service         ; Configuración base
        host_name                       penguin
        service_description             Actualizaciones
                check_command                   check_nrpe!check_apt!dummy
        }
```

Con ese fichero estaría definido el host ***penguin*** en el que se realizarían comprobaciones, con la frecuencia por defecto de ***Icinga***, para el espacio en disco, la carga del servidor y las actualizaciones disponibles.

Por otra parte, en el fichero ***/etc/icinga/objects/hostgroupsinciga.cfg*** se ha agregado como servidor ***SSH*** al host ***penguin***, quedando la sección correspondiente del siguiente modo:

```
# A list of your ssh-accessible servers
define hostgroup {
        hostgroup_name  ssh-servers
                alias           SSH servers
                members         localhost,penguin
        }
```

La intención es que en esa máquina se realicen comprobaciones sobre el servicio ***SSH.*** Para ello está definido el servicio ***SSH*** en el fichero ***/etc/icinga/servicesinciga.cfg*** para el grupo de servidores ***SSH,*** por lo que en la máquina también ***penguin*** se monitorizará.

```
# check that ssh services are running
define service {
        hostgroup_name                  ssh-servers
        service_description             SSH
        check_command                   check_ssh
        use                             generic-service
        notification_interval           0 ; set > 0 if you want to be renotified
}
```

Además de SSH, para la máquina ***penguin*** se monitorizarán las actualizaciones del sistema, carga, espacio en disco y disponibilidad del host mediante ***ping.*** Al menos eso es lo que debería ocurrir, pero aún es necesario instalar el servidor ***NRPE***, ***Nagios Remóte Plugin Executor***, en cada máquina monitorizada y el ***plugin*** en el servidor que ejecuta ***Icinga.***

Para instalar y configurar el servidor en la máquina***penguin*** se realiza lo siguiente:

```
apt-get install nagios-nrpe-server
```

Su fichero de configuración principal es ***/etc/nagios/nrpe.cfg*** y en él deben permitirse las conexiones desde la IP del servidor ***Icinga***, así como el pase de parámetros por parte del mismo. El fichero de configuración tendría un aspecto como el que sigue:

```
log_facility=daemon
pid_file=/var/run/nagios/nrpe.pid
server_port=5666
nrpe_user=nagios
nrpe_group=nagios
allowed_hosts=127.0.0.1,icinga.informatica64.com

dont_blame_nrpe=1
debug=0
command_timeout=60
connection_timeout=300
command[check_load]=/usr/lib/nagios/plugins/check_load -w 15,10,5 -c 30,25,20
command[check_all_disks]=/usr/lib/nagios/plugins/check_disk -w '20%' -c '10%'
-e
command[check_apt]=/usr/lib/nagios/plugins/check_apt
include=/etc/nagios/nrpe_local.cfg
include_dir=/etc/nagios/nrpe.d/
```

Como se puede observar, se han definido de forma adicional los comandos necesarios para adquirir la información deseada del host y pasar el resultado mediante el uso de ***check nrpe*** en la configuración del servidor de ***Icinga.*** Es decir, en la configuración de ***Icinga*** se utiliza el ***check command check_ nrpe*** pasando como parámetro, utilizando el símbolo de exclamación, cada uno de los comandos definidos en la configuración ***NRPE*** del host remoto.

Para que toda la configuración anterior funcionase sin problemas tan solo faltaría instalar en el servidor de ***Icinga*** el cliente ***NRPE*** con el paquete ***nagios-nrpe-plugin.***

Una vez recargada la configuración el resultado sería el que se puede apreciar en la imagen de la página siguiente.

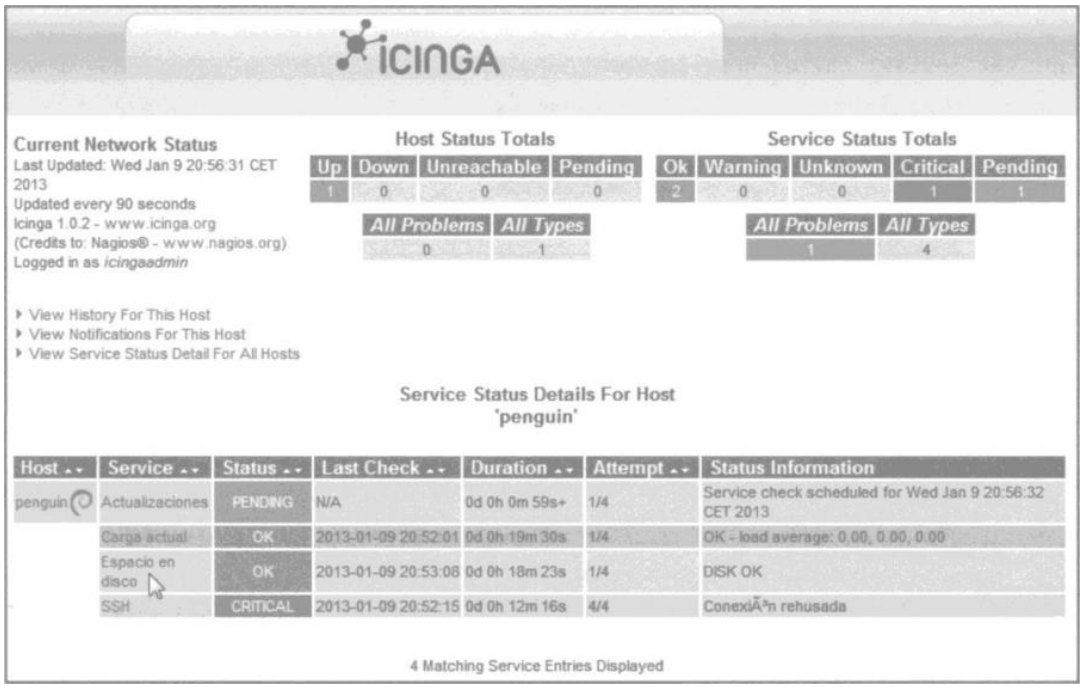


Fig 3.11: Estado de la máquina remota.

Adicionalmente se han generado notificaciones para las alertas encontradas en los sistemas monitorizados. Todo ello por defecto gracias a los grupos de notificación configurados por ***Icinga.*** La salida del buzón de correos del usuario administrador es la siguiente:

```
Message 9:
From nagios@squeetest Wed Jan 09 22:31:16 2013
Envelope-to: root@localhost
Delivery-date: Wed, 09 Jan 2013 22:31:16 +0100
Date: Wed, 09 Jan 2013 22:31:16 +0100
To: root@localhost
Subject: ** PROBLEM Service Alert: penguin/Espacio en disco is CRITICAL **
From: nagios@squeetest

***** Icinga *****

Notification Type: PROBLEM
```

```
Service: Espacio en disco
Host: penguin
Address: penguin.informatica64.com
State: CRITICAL

Date/Time: Wed Jan 9 22:31:16 CET 2013

Additional Info:

DISK CRITICAL - free space: / 272 MB (8% inode=81%):
```

Se puede observar como el espacio del servidor remoto monitorizado es crítico y se ha notificado por correo sobre el problema.

Capítulo IV
Protección de la red interna

En el capítulo anterior se estudió la protección del área perimetral explicando el uso de tres tipos de herramientas destinadas para ello. A pesar de ello, no por el hecho de haber establecido protecciones en la capa perimetral de una red significa que los servidores, puestos de usuario y los propios usuarios estén a salvo de ataques. Ahora llega el tumo de establecer algunas medidas de protección para la red intema, zona en la que en la mayoría de las ocasiones las brechas de seguridad o ataques vienen dados por suplantaciones de identidad. Se expondrán algunos ataques comunes, así como las contramedidas más utilizadas para su mitigación.

1. Spoofing o suplantación de identidad

En muchos casos se confía en que los usuarios de la red van a realizar un uso correcto de la misma. De forma voluntaria o involuntaria son dichos usuarios los que en numerosas ocasiones ponen en jaque la seguridad e integridad de un entorno. La mayoría de los ataques que se pueden realizar en una red interna a nivel de comunicación tienen como objetivo establecer un entorno ***M1TM***, ***Man In The Middle***, para poder así recuperar información sensible, realizar un ***DoS***, ***Denial of Service***, etc.

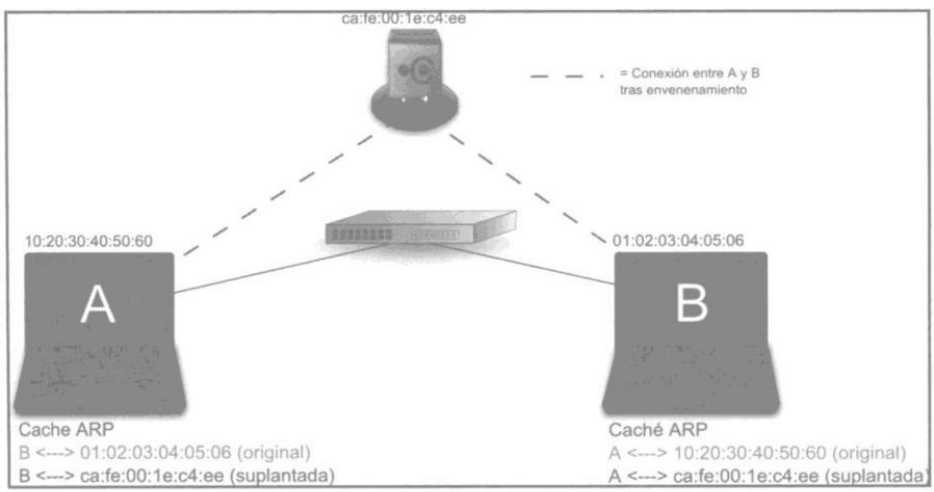


Fig 4.01: Entorno MITM entre dos máquinas.

ARP Poisoning

Este ataque consiste, como su nombre indica, en envenenar las tablas de cache ***ARP*** de las máquinas víctimas. Envenenando la caché ***ARP*** es posible conseguir un entorno ***MITM.*** Es decir, se trata de hacer creer a las máquinas víctimas que la máquina destino con la que desean comunicarse es la máquina del atacante. La elección de las máquinas víctimas siempre está a elección del atacante dependiendo de sus intereses. Habitualmente el escenario más elegido por un atacante consiste en situarse entre la puerta de enlace que da salida hacia internet o bien entre el ***DNS*** para poder suplantar las direcciones consultadas por la víctima.

A pesar de que principalmente el envenenamiento se utiliza para desplegar ataques, en ocasiones resulta necesario realizar este tipo de acciones cuando se desea auditar el tráfico de alguna máquina en concreto. Normalmente suele darse este caso en entornos pequeños o aislados donde no existe electrónica de red gestionable para configurar un ***port-mirroring*** por ejemplo.

Ejemplo de MITM entre el Gateway y el Cliente

Una vez que se han localizado las víctimas, tanto el ***gateway*** como el cliente, el atacante dispone de diversas herramientas para realizar su ataque. Todo depende de la plataforma utilizada por el atacante, necesidades, etc. En el caso de este ejemplo se opta por la utilización de las herramientas de la suite ***dsniff.*** En Debían es tan sencillo como instalar el paquete con el mismo nombre desde los repositorios oficiales. En este caso la herramienta utilizada para desplegar el ataque es ***arpspoof.***

Siguiendo la nomenclatura de la figura mostrada para MITM mostrada anteriormente, los datos de las máquinas son los siguientes:

- Máquina^, ***router*** o ***gateway.*** Dirección IP 192.168.87.1 con ***MAC*** D8:5D:4C:A4:FA:50.
- Máquina ***B***, máquina cliente o víctima. Dirección IP 192.168.87.56 con ***MAC*** 08:00:27: Al :2E:26.
- Máquina C, máquina atacante. Dirección IP 192.168.87.55 con ***MAC*** 08:00:27:E3:7A:7A

La tabla ARP de la víctima, antes de realizar el ataque tendrá el siguiente aspecto.

```
root@squeetest:~# arp -an
? (192.168.87.1) at d8:5d:4c:a4:fa:50 [ether] on eth0
```

Fig 4.02: Caché ARP de la víctima antes del ataque.

En la máquina atacante se realizaría el ataque del siguiente modo.

```
# arpspoof -i eth0 -t 192.168.87.56 192.168.87.1 &>/dev/null &
# arpspoof -i eth0 -t 192.168.87.1 192.168.87.56 &>/dev/null &
```

Se trata de la misma orden pero invirtiendo los parámetros para que el envenenamiento sea efectivo tanto en el ***gateway*** como en la máquina víctima. Así se podrá esnifar contenido en ambos sentidos.

Es decir, el tráfico saliente de la máquina víctima hacia Internet, así como el entrante de Internet será interceptado por la máquina atacante.

En este preciso instante, la máquina víctima seguiría saliendo a Internet por la puerta de enlace 192.168.87.1, pero sin embargo según la cache ***ARP*** esa IP ahora corresponde es la máquina atacante por su ***MAC*** asociada. Puede visualizarse en la imagen siguiente.

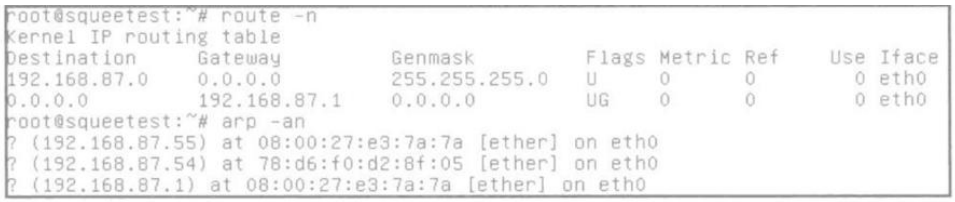


Fig 4.03: Caché ARP después del ataque.

Realizando solamente los pasos anteriores se habría conseguido un ***DoS,*** dado que la máquina atacante no es capaz de enrutar paquetes. Para ello, como se aprendió en el capítulo anterior, debe activarse el bit de enrutamiento. Por ejemplo con:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Ejemplo de MITM entre el DNS y el Cliente

Se trata de otros de los entornos típicos de un ***MITM*** en una red interna. Nuevamente es posible realizar un ***DoS*** con este ataque, pero generalmente se realiza con fines más peligrosos. Imagina lo que pasaría si se pudiera manipular un respuesta ***DNS*** para una solicitud de la web de un importante banco. El atacante podría clonar la web de dicho banco o abrir la original con un ***iframe*** y modificándola de tal modo que pudiera conseguir las credenciales del usuario. Las herramientas a utilizar en este caso son ***arpspoof*** para realizar el ataque ***MITM*** y ***dnsspoof*** para suplantar las respuestas ***DNS*** a las peticiones realizadas por la máquina víctima. Ambas herramientas pertenecen al paquete ***dsniff.***

Como en el caso anterior se sigue la nomenclatura del esquema inicial, pero en este caso los datos son los siguientes:

- Máquina^, servidor DNS. Dirección IP 192.168.87.1 con ***MAC*** D8:5D:4C:A4:FA:50.
- Máquina ***B***, máquina cliente o víctima. Dirección IP 192.168.87.56 con ***MAC*** 08:00:27:A1:2E:26.
- Máquina C, máquina atacante. Dirección IP 192.168.87.55 con ***MAC*** 08:00:27:E3:7A:7A

En el caso de este ejemplo el servidor ***DNS*** y el ***gateway*** o ***router*** que da salida hacia internet están configurados en el mismo dispositivo, por lo que se aprovecha el ***MITM*** conseguido en el ejemplo anterior. En caso de que el ***DNS*** se encuentre en una máquina diferente habría que realizar el ***MITM*** entre la máquina víctima y el servidor ***DNS*** con ***arpspoof.***

Se comprueba que la máquina cliente dispone de resolución ***DNS*** con la aplicación ***dig.*** En este instante, antes de realizar el ataque con ***dnsspoof \a.*** dirección resuelta es la real del nombre a resolver. Por ejemplo, resolviendo www.informatica64.com se obtendría la siguiente información.

```
root@squeetest:~# dig www.informatica64.com

; <<>> DiG 9.7.3 <<>> www.informatica64.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21521
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.informatica64.com.          IN      A

;; ANSWER SECTION:
www.informatica64.com.  2020    IN      A       80.81.106.147

;; Query time: 44 msec
;; SERVER: 192.168.87.1#53(192.168.87.1)
;; WHEN: Sun Jan 20 16:07:50 2013
;; MSG SIZE  rcvd: 55
```

Fig 4.04: Resolución de la dirección *www.informatica64.com* antes del ataque.

Uno de los parámetros que necesita ***dnsspoof*** para su funcionamiento es la ruta hacia un fichero con formato hosts. En este caso, como se está probando con el dominio de ***informatica64.com***, el contenido del fichero será el siguiente.

```
10.1.2.3        *.informatica64.com
```

Es decir, todos los subdominios de ***informatica64.com*** serán resueltos hacia la IP ***10.1.2.3.*** Una IP en la que el atacante podría disponer de un servidor web con un aplicación especialmente manipulada para el robo de credenciales. La aplicación ***dnsspoof*** se lanzará en este caso del siguiente modo en la máquina atacante:

```
# dnsspoof -i eth0 -f listado.hosts
```

Siendo ***ethO*** la interfaz de red utilizada y ***listado.hosts*** el fichero que contiene el contenido de la resolución mencionada más arriba. Si la máquina víctima tratase de resolver la dirección www. ***informatica64.com*** tras el ataque, obtendría el siguiente resultado.

```
root@squeetest:~# dig www.informatica64.com

; <<>> DiG 9.7.3 <<>> www.informatica64.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 4044
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.informatica64.com.          IN      A

;; ANSWER SECTION:
www.informatica64.com.  60      IN      A       10.1.2.3

;; Query time: 12 msec
;; SERVER: 192.168.87.1#53(192.168.87.1)
;; WHEN: Sun Jan 20 16:21:14 2013
;; MSG SIZE  rcvd: 55
```

Fig 4.05: Resolución de la dirección *www.informatica64.com* después del ataque.

Protección contra ARP Poisoning

Queda comprobado que un ataque ***MITM*** aprovechando la funcionalidad ***ARP*** es muy peligroso si se lleva a cabo de forma transparente consiguiendo que el usuario no sospeche que está siendo atacado.

Entradas ARP estáticas

La más obvia de ellas es la utilización de entradas ***ARP*** estáticas en las máquinas cliente para que estas no puedan ser atacadas con éxito. Se trata de un método muy simple y fácil de llevar a cabo solo si se está en un entorno pequeño. Cuando existen muchas máquinas puede resultar inviable realizar esta operación. A pesar de ello resulta efectivo, por lo que se muestra como llevar a cabo esta operación para proteger a una máquina cliente cuando envía sus paquetes hacia la puerta de enlace o ***router.***

Con lo que se crearía una entrada estática en la caché ***ARP*** de la máquina y no podría ser sustituida por En la máquina cliente habría que ejecutar en cada arranque la siguiente orden:

```
# arp -s 192.168.87.1 d8:5d:4c:a4:fa:50
```

Otra opción para establecer entradas estáticas en la caché ***ARP*** pasa por utilizar un fichero con una relación de direcciones ***MAC*** e IP/nombre. Siguiendo con el ejemplo, dicho fichero podría contener

lo siguiente:

```
d8:5d:4c:a4:fa:50     192.168.87.1
# <otra MAC>          <otra ip>
```

ataques realizados con ***arpspoof*** por ejemplo.

En cada arranque habría que ejecutar la orden ***arp -f <fichero>.*** En caso de no indicar el fichero se leería el contenido del fichero ***/etc/ethers.***

Monitorización de ARP

Como se comentaba con anterioridad, no siempre resulta posible o viable mantener entradas estáticas ***ARP*** en las máquinas de un entorno. Si fuese posible, sin duda se estaría evitando el ataque pero el administrador de la red puede que no supiera que está habiendo intentos de ataques ***ARP.*** Es por ello que existen aplicaciones que se ejecutan como demonios en las máquinas de una red que detectan si se está sufriendo un ataque ***ARP Poisoning***, evitándolo y notificándolo al administrador de la red.

En este caso se opta por el uso de ***arpwatch***, una herramienta muy conocida en entornos ***Linux.*** Es ***OpenSource*** y está disponible para su instalación desde los repositorios oficiales de muchas distribuciones.

Una vez instalada y con la configuración por defecto, detectará los ataques ***ARP*** y hará que resulten totalmente inefectivos. Siguiendo el mismo ejemplo de ***MITM***, las siguientes imágenes muestran cómo ***arpwatch*** ha detectado una ataque.

```
Jan 20 17:20:05 squeetest arpwatch: new station 192.168.87.55 08:00:27:e3:7a:7a
eth0
Jan 20 17:20:06 squeetest arpwatch: changed ethernet address 192.168.87.1 08:00:
27:e3:7a:7a (d8:5d:4c:a4:fa:50) eth0
Jan 20 17:20:10 squeetest arpwatch: flip flop 192.168.87.1 d8:5d:4c:a4:fa:50 (08
:00:27:e3:7a:7a) eth0
Jan 20 17:20:12 squeetest arpwatch: flip flop 192.168.87.1 08:00:27:e3:7a:7a (d8
:5d:4c:a4:fa:50) eth0
Jan 20 17:20:22 squeetest arpwatch: flip flop 192.168.87.1 d8:5d:4c:a4:fa:50 (08
:00:27:e3:7a:7a) eth0
```

Fig 4.06: Log de *arpwatch* con el ataque ARP.

Igualmente se ha generado un correo electrónico especificando que se ha producido un ataque e indicando a qué host ha afectado, así como las direcciones ***MAC*** suplantada y original.

```
Message 21:
From arpwatch@squeetest Sun Jan 20 17:20:06 2013
Envelope-to: root@squeetest
Delivery-date: Sun, 20 Jan 2013 17:20:06 +0100
Date: Sun, 20 Jan 2013 17:20:06 +0100
From: arpwatch@squeetest (Arpwatch squeetest)
To: root@squeetest
Subject: changed ethernet address (192.168.87.1) eth0

            hostname: <unknown>
          ip address: 192.168.87.1
           interface: eth0
    ethernet address: 08:00:27:e3:7a:7a
     ethernet vendor: CADMUS COMPUTER SYSTEMS
old ethernet address: d8:5d:4c:a4:fa:50
 old ethernet vendor: <unknown>
           timestamp: Sunday, January 20, 2013 17:20:06 +0100
  previous timestamp: Sunday, January 20, 2013 17:19:15 +0100
```

Fig 4.07: Correo electrónico generado por *arpwatch*.

Utilización del fichero /etc/hosts

Si las entradas ***ARP*** estáticas resultaban difíciles de mantener, es posible que establecer una equivalencia IP - Nombre sea aún más complicado. Esta es una contramedida adicional cuando lo anterior ha fallado para al menos proteger las respuestas ***DNS*** de algunos sitios o zonas.

Si por ejemplo se suele consultar una zona privada de www.informatica64.com muy a menudo y se sabe con certeza que la dirección IP de ese recurso no va a ser modificada, es posible establecer esa resolución forma estática en la máquina cliente. Se podría añadir la siguiente entrada al fichero ***/etc/hosts*** de la máquina ***Linux*** cliente:

```
80.81.106.147  www.informatica64.com informatica64.com
```

Por lo que a pesar de que la máquina sea atacada con ***DNS Spoofing***, la dirección ***www. informatica64. com*** siempre será resuelta con su dirección real.

DHCP Spoofing

También conocido como ***Rogue DHCP*** se trata de un concepto muy sencillo. Este ataque consiste en implementar un servidor ***DHCP*** alternativo con la finalidad de asignar direcciones IP a aquellas máquinas que las adquieran de forma automática. Obviamente el servidor ***DHCP*** estará configurado según los intereses del atacante, estableciendo valores como puerta de enlace, servidores ***DNS***, etc. De ese modo la víctima verá su tráfico comprometido y podrá ser esnifado e incluso manipulado por el atacante.

Ejemplo de Rogue DHCP

Si se pudiera visualizar el direccionamiento IP obtenido por la máquina del cliente o víctima, con el servidor DHCP lícito su reserva tendría un aspecto como el siguiente:

```
lease {
  interface "eth0";
  fixed-address 192.168.87.52;
  option subnet-mask 255.255.255.0;
  option routers 192.168.87.1;
  option dhcp-lease-time 7200;
  option dhcp-message-type 5;
  option domain-name-servers 192.168.87.1,8.8.8.8;
  option dhcp-server-identifier 192.168.87.1;
  option domain-name "informatica64.com";
  renew 2 2013/01/22 12:51:22;
  rebind 2 2013/01/22 13:40:39;
  expire 2 2013/01/22 13:55:39;
}
```

Es decir, la víctima dispone de esa reserva en la que su puerta de enlace es ***192.168.87.1*** y sus servidores ***DNS*** son ***192.168.87.1*** y ***8.8.8.8.***

Si el atacante quisiera configurar un servidor ***DHCP*** respetando la subred original del cliente pero modificando la puerta de enlace y los servidores ***DNS*** podría configurar el servidor ***DHCP*** falso o alternativo del siguiente modo.

```
# La red Maligna
subnet 192.168.87.0 netmask 255.255.255.0 {
  range 192.168.87.130 192.168.87.140;
  option domain-name-servers 192.168.87.55;
```

```
  option domain-name "informatica64.com";
  option routers 192.168.87.55;
  default-lease-time 600;
  max-lease-time 7200;
}
```

El fragmento anterior corresponde a una configuración básica del servidor DHCP ***isc-dhcp-server.*** Se puede observar cómo se han modificado los parámetros ***option-routers*** y ***domain-name-servers*** de modo que la víctima utilice dichos parámetros y el atacante pueda hacerse con el tráfico de la víctima.

En la máquina del cliente o víctima se obtendría el siguiente resultado para la reserva DHCP si hubiera obtenido la configuración del servidor DHCP suplantado.

```
lease {
  interface "eth0";
  fixed-address 192.168.87.130;
  option subnet-mask 255.255.255.0;
  option routers 192.168.87.55;
  option dhcp-lease-time 600;
  option dhcp-message-type 5;
  option domain-name-servers 192.168.87.55;
  option dhcp-server-identifier 192.168.87.55;
  option domain-name "informatica64.com";
  renew 2 2013/01/22 12:21:43;
  rebind 2 2013/01/22 12:25:56;
  expire 2 2013/01/22 12:27:11;
}
```

Protección contra DHCP Spoofing

Puede parecer sencillo proteger a una máquina frente a servidores ***DHCP*** ilícitos, pero no es así. Uno de los principales métodos que se pueden idear es la utilización de ***iptables*** con la finalidad de permitir respuestas, a las peticiones ***DHCP*** lanzadas por la propia máquina, desde los servidores ***DHCP*** que se especifiquen. Es decir, bloquear las respuestas desde los puertos ***67/udp*** que provengan de máquinas que no sean el servidor ***DHCP*** lícito, corporativo, etc.

Lo que ocurre es que lo anterior no funciona en absoluto. A pesar de establecer la política por defecto a ***DROP*** en ***iptables*** para todas las interfaces de red de la máquina, esta continuaría adquiriendo direcciones IP de cualquier máquina que las ofrezca a través de su servicio ***DHCP.*** El motivo de

este suceso reside en que la comunicación ***DHCP*** se realiza a través de los llamados ***sockets raw,*** que por diseño son procesados antes que el posible filtrado que pueda realizar ***iptables.*** Son algunas aplicaciones o protocolos de enrutamiento los que soportan este tipo de transmisión, y uno de ellos ha resultado ser ***DHCP*** y más en concreto el servidor montado anteriormente por el atacante.

A pesar de ello, es posible tomar ciertas medidas y precauciones para tratar de mitigar los ataques ***Rogue DHCP.*** Por supuesto, siempre que sea posible la configuración estática del direccionamiento de algunas máquinas es recomendable realizarlo de tal modo en lugar de utilizar reservas estáticas en los servidores ***DHCP***, etc.

Configuración de los clientes DHCP

En los clientes ***DHCP***, ***dhclient*** de las máquinas ***Linux*** y más en concreto en el fichero ***/etc/dhcp/dhclient.conf,*** existe una directiva con la que se puede crear una lista negra de servidores ***DHCP*** no lícitos y que por lo tanto son perjudiciales para el entorno de red.

Sin lugar a dudas es una funcionalidad que no proporciona una protección proactiva y puede que en el momento que se detecte un posible ataque sea demasiado tarde para actuar. A pesar de ello no está de más que se configure en caso de urgencia.

Si por ejemplo se quisiese bloquear la dirección IP ***192.168.87.55,*** pues se trata de un servidor ***DHCP*** malicioso, debería agregarse al fichero ***/etc/dhcp/dhclient. confia,*** siguiente línea:

```
reject 192.168.87.55;
```

Una vez salvada dicha configuración, cada vez que el cliente tuviese que adquirir direccionamiento automático recibiría mensajes como los siguientes.

```
root@test:~# dhclient -v br0
Internet Systems Consortium DHCP Client 4.2.2
Copyright 2004-2011 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/br0/08:00:27:6f:0b:73
Sending on   LPF/br0/08:00:27:6f:0b:73
Sending on   Socket/fallback
DHCPDISCOVER on br0 to 255.255.255.255 port 67 interval 7
DHCPOFFER from 192.168.87.55 rejected by rule 192.168.87.55 mask 255.255.255.255.
DHCPDISCOVER on br0 to 255.255.255.255 port 67 interval 12
DHCPOFFER from 192.168.87.55 rejected by rule 192.168.87.55 mask 255.255.255.255.
^C
```

Protección local de la máquina mediante ebtables

Anteriormente se habló de ***iptables***, qué es, para qué sirve, etc. De igual modo se ha mencionado al comenzar este punto que tratar de proteger una máquina frente a ***DHCP Spoofing*** mediante el uso de reglas ***iptables*** resultará en vano, por el modo de comunicación de ***DHCP.*** En lugar de utilizar ***iptables***, es posible utilizar su análogo para la capa 2.

La única pega reside en que solo es posible utilizarlo para dispositivos de tipo ***bridge***, por lo que filtrar en un sistema convencional conllevará crear un dispositivo ***bridge*** y asociarle la interfaz o interfaces de red deseadas. Todo ello puede realizarse con la herramienta ***brctl*** contenida en el paquete ***bridge-utils*** que se encuentra disponible en los repositorios oficiales. Una vez creado el dispositivo ***bridge***, que sería el dispositivo a configurar por ***DHCP***, se podría realizar un filtrado del siguiente modo:

```
# ebtables -A INPUT -p 0x0800 --ip-proto udp --ip-source ! 192.168.87.1 --ip-
source-port 67:68 -j DROP
```

En el caso de ejemplo se está indicando que todas las peticiones desde los puertos 67 y 68, protocolo IP ***UDP*** y protocolo IP ó 0x0800, provenientes de direcciones diferentes a la dirección IP ***192.168.87.1*** sean descartadas. Lógicamente esa dirección IP es la asociada a un servidor ***DHCP*** lícito.

Sin duda resulta un método bastante tedioso y poco eficiente para aplicar a todas las máquinas de un entorno corporativo, red ***SOHO***, doméstica, etc. De igual modo ocurre con el bloqueo utilizando directivas en el cliente ***dhclient.*** Normalmente se suelen utilizar configuraciones de electrónica de red gestionable para estos menesteres. En concreto se utiliza ***DHCP Snooping*** para los dispositivos que dispongan de dicha funcionalidad. Básicamente permite indicar por qué boca física del ***switch***, qué direcciones ***MAC*** o IP pueden ofrecer direcciones mediante ***DHCP.*** Unido a configuraciones ***VLAN*** e incluso monitorización de la red, se reduciría el riesgo de ataques por ***Rogue DHCP*** de una forma más eficiente.

ICMP Redirect

El protocolo ***ICMP***, ***Internet Control Message Protocol***, es utilizado con diferentes finalidades para el control y notificación de errores del protocolo IP. Entre los tipos de mensajes de ***ICMP*** más conocidos se encuentran los usados para completar un***ping***, ***icmp request*** e ***icmp reply***, o por ejemplo el ***time exceeded*** cuando el ***TTL*** de un paquete ha expirado en tránsito, etc.

Sin embargo, existe un tipo de mensaje ***ICMP*** que puede resultar muy atractivo y de gran utilidad a un atacante a la hora de desplegar un ataque. Se trata de ***icmp redirect***, un mensaje ***ICMP*** de ***tipo 5*** que permite modificar la tabla de ruta de una máquina víctima con la finalidad de redirigirla hacia otros destinos en beneficio del atacante. Es posible hacerlo para una red entera o bien para un host determinado. Obviamente es el uso que le daría un atacante, pero existen este tipo de mensajes ***ICMP*** para mejorar la conectividad en redes congestionadas, respetar la latencia de servicios en función del ***ToS*** o ***Type of Service*** encontrando nuevas rutas, etc.

Ejemplo de ICMP Redirect malicioso

Existen varias herramientas con las que un atacante puede realizar un ataque ***icmp redirect***, pero la que se muestra en este ejemplo es ***hping3.*** Dicha herramienta está disponible en los repositorios oficiales de las distribuciones y su instalación es extremadamente sencilla.

En este ejemplo el atacante va a redirigir los paquetes hacia la IP destino ***8.8.8.8*** desde la IP origen de la máquina víctima ***192.168.87.56***, para que pasen por la IP asignada a la máquina del atacante que será la ***192.168.87.53.*** Es decir, en lugar de que las peticiones hacia la dirección indicada salgan directamente por la IP de la puerta de enlace de la víctima, pasarán en su lugar por la ***192.168.87.55.*** Para ello, el atacante lanzaría el ataque del siguiente modo:

```
# hping3 -c 1 -C 5 -K 1 -a 192.168.87.1 --icmp-ipdst 8.8.8.8 --icmp-gw 192.168.87.53
--icmp-ipsrc 192.168.87.56 192.168.87.56
```

La IP ***192.168.87.1*** es la IP de la puerta de enlace que se está suplantando para que la máquina víctima acepte la redirección. De igual modo, el atacante debe haber configurado la máquina en modo ***router***, incluso con una regla de salida ***NAT*** para enmascarar los paquetes y poder visualizar las respuestas del host destino, además de deshabilitar el envío de redirecciones. Resulta necesario realizar ese último paso para que la máquina del atacante no envíe un ***icmp redirect*** a la víctima indicando que el mejor camino para llegar a la ***8.8.8.8*** es la IP ***192.168.87.1.***

A partir de este momento, las comunicaciones entre la IP ***8.8.8.8*** y la máquina del cliente estarían siendo interceptadas. Realizando un ***ping*** hacia la dirección ***8.8.8.8*** desde la máquina víctima podría visualizarse una diferencia en el TTL cuando la ruta se ha modificado.

```
root@squeetest:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=48 time=64.3 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=48 time=48.1 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=48 time=48.2 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=48 time=51.9 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=48 time=72.6 ms
64 bytes from 8.8.8.8: icmp_req=6 ttl=48 time=49.0 ms
64 bytes from 8.8.8.8: icmp_req=7 ttl=48 time=60.3 ms
64 bytes from 8.8.8.8: icmp_req=8 ttl=48 time=47.6 ms
64 byt                                              7.5 ms
```

Fig 4.08: *Ping* desde la víctima a 8.8.8.8 sin *icmp_redirect*.

```
root@squeetest:~# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=47 time=76.5 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=47 time=48.4 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=47 time=49.2 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=47 time=47.8 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=47 time=49.0 ms
64 bytes from 8.8.8.8: icmp_req=6 ttl=47 time=49.0 ms
64 bytes from 8.8.8.8: icmp_req=7 ttl=47 time=47.7 ms
64 bytes from 8.8.8.8: icmp_req=8 ttl=47 time=47.5 ms
64 bytes from 8.8.8.8: icmp_req=9 ttl=47 time=64.2 ms
```

Fig 4.09: *Ping* desde la víctima a 8.8.8.8 con *icmp_redirect*.

Protección frente a ICMP Redirect

Por suerte es posible configurar una máquina de forma local para evitar que los mensajes ***icmp redirect*** surtan efecto. Consiste en modificar un fichero con 0 ó 1:

```
/proc/sys/net/ipv4/conf/all/accept_redirects
```

Con valor 1 se aceptarán redirecciones por ***ICMP*** y serán descartadas cuando el valor sea 0. En la mayoría de las distribuciones de hoy en día dicho valor está establecido a 0 por lo que las máquinas estarán protegidas por defecto. A pesar de ello es necesario conocer la configuración de este parámetro para establecer un grado más de seguridad en entornos ***Linux.***

Por supuesto, todos estos ataques que se han comentado afectan a redes IPv4. Como se comentó en el capítulo anterior, en caso de no utilizar IPvó es mejor desactivarlo por completo para evitar posibles ataques propiciados por configuraciones por defecto. Entre ellos están ***Rogue DHCPvó***, ***SLAAC,*** envenenamiento de vecinos, etc. Se ha de recordar que por diseño, los sistemas operativos modernos siempre darán prioridad a IPvó cuando sea posible su uso en una red por lo que un atacante podría aprovecharse de ese comportamiento.

2. VLAN

Cuando existen redes formadas por un gran número de máquinas resulta necesario realizar una correcta división o segmentación de la misma. Ello evitará problemas de rendimiento, seguridad y gestión de la propia red entre otros aspectos. Puede que, en la mayoría de los casos esto requiera de un elevado número de elementos de electrónica de red para llevarlo a cabo, routers para separación de redes, cortafuegos, interfaces de red, etc.

Una ***VLAN***, ***Virtual LAN,*** no es más que el concepto de una red virtual. Las ***VLAN*** se utilizan para crear diferentes redes lógicas a nivel de ***switch*** en los casos que estos permitan dichas configuraciones. Existen varios protocolos para implementar soluciones ***VLAN***, pero el estándar es ***802. lq*** que es el más extendido y por lo tanto será utilizado en este caso. Entre algunos de los usos que se le pueden dar a una ***VLAN*** se encuentran:

- Agrupaciones y división de equipos conectados a un mismo ***switch.***
- Separación de telefonía IP y los datos, así como su priorización.
- Aislamiento de zonas, restricciones, etc. Esencial para reducir la zona de impacto.

Y la pregunta es, qué tiene que ver todo esto con ***Linux*** si se está comentando que son los ***switches*** gestionables y con soporte para ***VLAN*** los que realizan estas funciones. Obviamente, para hacer que un equipo ***Linux*** se comporte como enrutador y cortafuegos entre las diferentes ***VLAN.***

En principio cabría pensar que para cada ***VLAN*** existente debería utilizarse una interfaz de red diferente y dedicada a cada una de dichas ***VLAN:***; pero en el caso de que existieran muchas ***VLAN***

resultaría costoso e incluso inabordable. Para evitar lo anterior es posible utilizar la técnica de ***trunking***, que no es más que hacer que una boca física del ***switch*** pueda transportar tráfico de múltiples ***VLAN.*** En esos casos, para diferenciar los tráficos de las diferentes ***VLAN*** se añaden 4 bytes a cada trama. En esos 4 bytes se agrega la siguiente información:

- Prioridad. Define la prioridad de la trama utilizando 3 bits. Acepta valores del 0 al 7, siendo 0 el valor por defecto. Dicho valor corresponde al mecanismo de entrega ***Best Effort.*** Para más información sobre este aspecto puede consultarse ***CoS***, ***Class of Service.*** No es en absoluto irrelevante teniendo en cuenta que con la técnica de ***trunking*** se comparte el mismo medio físico para el transporte de diferentes ***VLAN.***
- CFI. Se trata de un valor de un solo bit utilizado para indicar si se debe descartar la trama en caso de congestión.
- ID. Es la etiqueta que identifica a la ***VLAN.*** Su tamaño es de 12 bits y puede tener un valor comprendido entre el 0 y 4095. Inicialmente, en un ***switch*** desconfigurado todas las bocas se encuentran en la ***VLAN*** 1.
- Tipo. Información sobre el tipo de protocolo de red que contiene la trama. Su tamaño es de 32 bits. Un valor podría ser ***0x0800*** que equivale a IPv4 o bien ***0x86DD*** para IPv6.

En el caso de utilizar múltiples interfaces de red en una máquina ***Linux*** que hiciera de enrutador y/ ***firewall*** no sería necesaria la gestión de las ***VLAN*** como tales. Sin embargo, cuando se utiliza ***trunking*** resulta necesario utilizar determinadas herramientas y configuraciones en una máquina ***Linux*** para interactuar con diferentes ***VLAN*** utilizando una sola interfaz de red. Realmente se terminará trabajando con una serie de interfaces virtuales que corresponderán a cada una de las VLAN. Es lo que recibe el nombre de interfaz ***tagged***, puesto que pertenece a más de una ***VLAN.***

Configuración de VLAN en Linux

Para empezar a utilizar y configurar ***VLAN*** en ***Linux*** es necesario averiguar si la máquina ***Linux*** es capaz de trabajar con ***VLAN*** verificando si dispone del módulo necesario.

```
# modprobe 8021q && OK
```

Si lo anterior devuelve OK, la máquina estará preparada para trabajar con ***VLAN.*** Ahora solo falta que en cada arranque se cargue en memoria el módulo. Se puede editar el fichero ***/etc/modules*** y agregar el módulo o bien ejecutar la siguiente orden:

```
# echo 8021q >> /etc/modules
```

A partir de este momento se es libre de elegir cómo se desean configurar las diferentes interfaces para ***VLAN.*** Es posible utilizar el conocido comando ***ip*** para crear interfaces de red virtuales y configurarla. En este caso se hace uso de la aplicación ***vconfig.*** En el caso de Debian se puede encontrar en el paquete ***vían***, instalable desde los repositorios oficiales.

Una vez instalado, es posible agregar la definición de las interfaces virtuales en el fichero ***/etc/network/interfaces*** o bien lanzar la aplicación ***vconfig*** con los parámetros adecuados. Antes de empezar es necesario saber que la nomenclatura de las interfaces virtuales se suele establecer a ***<interfazFísica>.<vlanID>*** y es como se realiza por defecto. Por ejemplo, para poder gestionar las ***VLAN*** 10 y 20, dependientes de la interfaz física ***ethl,*** quedaría un ***/etc/network/interfaces*** como el siguiente.

```
auto lo
iface lo inet loopback

allow-hotplug eth0
iface eth0 inet dhcp

auto eth1 eth1.10 eth1.20
iface eth1 inet static              #Interfaz física para las VLAN
        address 172.16.0.1
        netmask 255.255.255.0

iface eth1.10 inet static           #VLAN 10
        address 10.0.0.1
        netmask 255.255.255.0

iface eth1.20 inet static           #VLAN 20
        address 10.0.1.1
        netmask 255.255.255.0
```

Es posible visualizar el estado con el ***ifconfig***, ***ip link*** o incluso desde el fichero ***/proc/net/vlan/config.*** Por otra parte se podrá establecer el valor para la prioridad utilizando ***vconfig.***

Por supuesto, esta es una configuración básica con la que empezar en un entorno ideado para ***VLAN.*** A partir de este momento faltarían configurar rutas, reglas ***iptables***, ***QoS***, etc.

3. IPsec

Internet Protocol Security, IPsec, es como se define a un conjunto específico de protocolos y mecanismos destinados a proteger la capa 3 o nivel de red. El motivo de su creación surgió por la necesidad de dotar de seguridad al protocolo IP, inseguro por diseño. Inicialmente fue ideado para

su uso exclusivo con IPv6, pero más adelante fue adaptado para trabajar con IPv4 aunque con ciertas limitaciones como se verá posteriormente. En este último, su uso es opcional pero sin embargo en el caso de IPv6 es obligatorio. ***IPsec*** puede ser configurado de diversos modos y dependiendo de dichas configuraciones y de cada entorno se pueden obtener los siguientes beneficios con su utilización.

- Cifrado. Es posible configurar ***IPsec*** para que el tráfico sea cifrado y solo los extremos implicados en la comunicación serán capaces de descifrar e interpretar dicho tráfico. Decir extremos resulta un tanto relativo, pues todo depende del modo en que se configure ***IPsec.***
- Autenticación e integridad. En ***IPsec*** es posible saber si un paquete ha sido modificado desde su origen, al menos los datos y parámetros que se utilizan para calcular el digest de verificación. Para realizar dicha operación ha sido necesario realizar una autenticación de las máquinas para establecer los parámetros criptográficos.
- Protección frente a ataques de tipo ***Replay.*** Este tipo de ataques consiste en reenviar o reinyectar código inicialmente lícito de forma maliciosa por parte de un atacante. Una de las partes susceptibles a este tipo de ataques es la negociación inicial de las claves. En ***IPsec*** se implementan mecanismos de secuencia y ventanas de tiempo que harán que los posibles paquetes capturados por un atacante no puedan volver a utilizarse.

Sobre el funcionamiento de IPsec

En este apartado se explicarán los aspectos más importantes sobre ***IPsec.*** Entre ellos se encuentran los conceptos básicos de una implementación ***IPsec***, los modos en los que puede trabajar, así como los protocolos que proporcionan los beneficios enumerados anteriormente.

Conceptos básicos

Antes de comenzar a explicar los siguientes términos, es necesario conocer que los modos de trabajo de ***IPsec*** son transporte y túnel. Por otra parte, los protocolos utilizados en ***IPsec*** son ***AH*** y/o ***ESP,*** destinados a la parte de criptografía para la protección de los datagramas. Para establecer los valores criptográficos existe el protocolo ***IKE***, si bien es cierto que pueden ser establecidos manualmente. Estos conceptos serán explicados posteriormente, pero es necesario al menos saber que existen para definir los siguientes términos.

- ***SA***, ***Security Association***, consiste en información que contiene la dirección IP origen y destino, el protocolo de ***IPsec*** utilizado, el ***SPI***, el algoritmo de cifrado o autenticación y una clave. Dicha información es almacenada en la llamada ***SAD, Security Association Database.*** Es decir, con la ***SA*** se define con qué protocolo se configura la conexión segura para unas direcciones en concreto. La base de datos puede ser mantenida de forma dinámica mediante el uso de ***IKE*** o bien de forma manual. Ambos casos se estudiarán con unos ejemplos posteriormente.
- ***SP***, ***Security Policy.*** Se trata de las definiciones de políticas para los paquetes. En este punto es donde se indica qué hacer con cada tipo de paquete. Se especifican las direcciones origen y destino, el protocolo para el que usar dicha política y la política en sí. Posteriormente se establece el protocolo utilizado, modo de trabajo, origen y destino si se trabaja en modo

túnel y por último el nivel de "severidad" a aplicar al tipo de paquete. Ese último parámetro decidirá si el paquete puede viajar a una máquina no configurada con ***IPsec***, es decir que solo se permitan conexiones seguras por ejemplo. Los paquetes que finalmente utilicen ***IPsec*** serán marcados con el ***SPI***, ***Security Parameter Index*** definido en la ***SAD.*** Las políticas o ***SP*** se almacenan en la ***SPD***, ***Security Policy Database.***

Modos de trabajo

Como se comentaba anteriormente, ***IPsec*** dispone de dos modos de trabajo, siendo posible trabajar en modo túnel o en modo transporte.

Modo túnel

Consiste en proteger todo el tráfico entre dos puntos que posteriormente se podrá repartir a otras máquinas. Todo depende del contenido protegido, si va destinado a la propia máquina que descifra ***IPsec*** o no. Es un caso muy típico en situaciones donde se desea proteger la comunicación entre dos redes; en estos casos se configura un túnel entre los dos ***routers*** que proporcionan comunicación a dichas redes.

En los casos de túnel los paquetes que se generan están encapsulando el paquete IP original, siendo necesario escribir una nueva cabecera IP, independientemente del protocolo utilizado para proteger dicho paquete. Se puede concluir por lo tanto que solo se protege el tráfico entre los extremos del túnel.

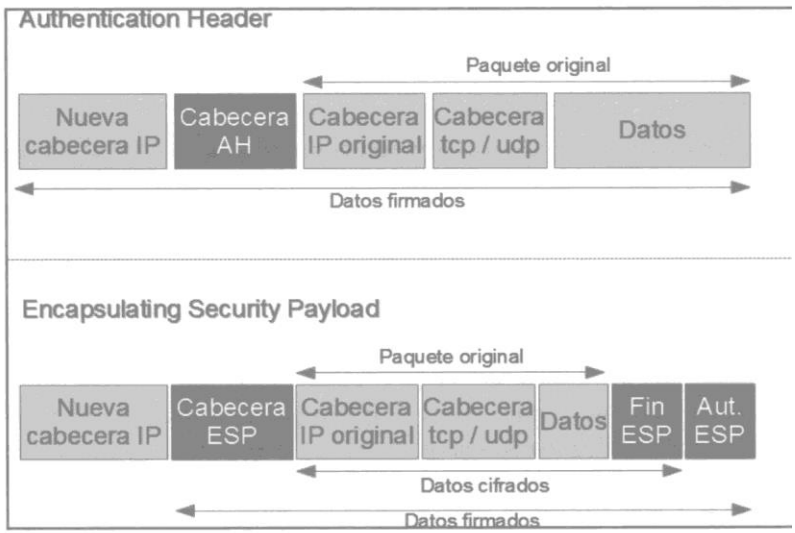


Fig 4.10: Modo túnel.

Modo transporte

Se protege el tráfico entre dos extremos o ***endpoints*** configurados para el uso de ***IPsec.*** Puede resultar aplicable por ejemplo a una conexión entre dos máquinas para el uso de escritorio remoto, XI1,

telnet, tráfico http, etc. Realmente es muy parecido al modo anterior pero ambos tienen enfoques diferentes. En este caso la cabecera IP original se mantiene y se añaden las cabeceras e información de AH o ESP al paquete que se está protegiendo.

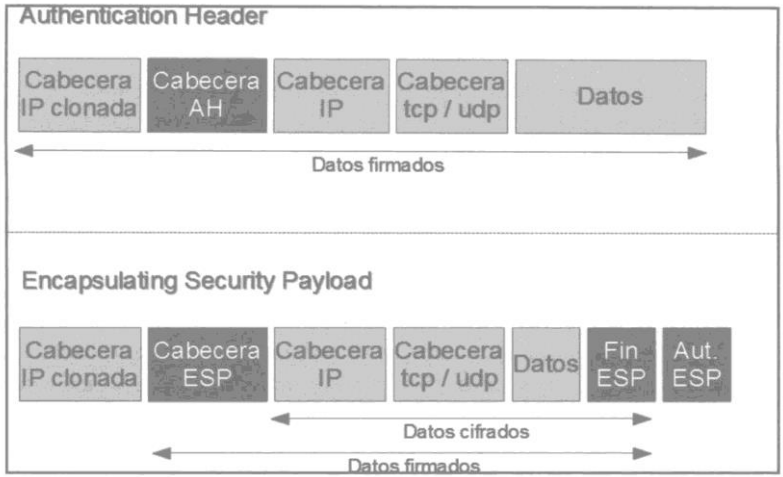


Fig 4.11: Modo transporte.

Protocolos principales utilizados en IPsec

En este punto se explicarán tres protocolos. Dos de ellos relacionados con la protección de los datos y el restante para el intercambio de claves en una conexión ***IPsec.***

AH, Authentication Header

El protocolo ***AH*** ofrece integridad al protocolo IP utilizando determinados algoritmos configurados por el usuario como pueden ser ***md5, shal***, ***sha256***, ***sha512***, etc. En concreto calcula un ***HMAC***, ***Hash Message Authentication Code*** de las partes que no deben sufrir ninguna modificación en el paquete como son direcciones IP, el ***payload*** IP con sus datos, encabezado de protocolo contenido, etc. Una vez calculados los datos se añade una cabecera ***AH*** al paquete IP original con la siguiente información.

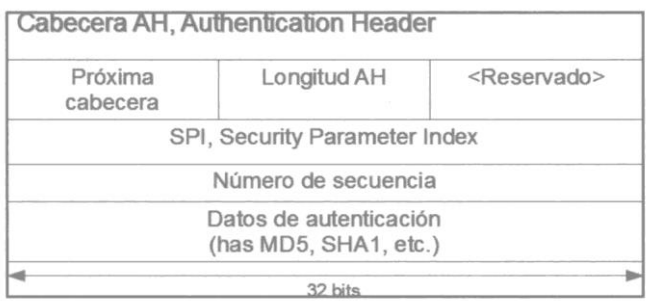


Fig 4.12: Estructura de una cabecera AH.

Entre los parámetros agregados se destacan los siguientes.

- ***Next Header.*** Establece el contenido de la siguiente cabecera del paquete protegido. En modo transporte, donde solo se protegen los datos tendría el valor del protocolo protegido. Por ejemplo UDP, TCP, etc. pero nunca IP. En el caso del modo túnel este campo tendría el valor de IP ya que se está protegiendo el paquete IP al completo.
- ***SPI***, ***Security Parameter Index*** especifica la asociación de seguridad necesaria para poder leer el contenido del paquete protegido. En definitiva, para poder desencapsularlo.
- ***HMAC*** consiste en la cadena generada para la protección del paquete. Cuando el otro extremo de la comunicación ***IPsec*** recibe un datagrama debe poder calcular nuevamente el ***HMAC*** y compararlo con el contenido en la cabecera ***AH*** para verificar si el datagrama protegido ha sido modificado o no.

Como se comentaba antes, ***AH*** protege partes supuestamente inmutables del datagrama IP como son las direcciones IP. El problema se plantea cuando se utiliza ***SNAT***, ya que es necesario modificar dichos datos para poder enrutar el paquete. Por lo tanto, en los casos donde es necesario utilizar ***SNAT*** no es posible utilizar ***AH.***

ESP, Encapsulated Security Payload

Este protocolo puede proporcionar tanto autenticación como privacidad en las comunicaciones IP. Por lo tanto, después de cifrar el datagrama original según el modo de ***IPsec*** utilizado, y calculado el ***HMAC***; se agregan las cabeceras ***ESP.*** Para calcular el ***HMAC*** tan solo se tiene en cuenta el ***payload*** del paquete, no se incluye la cabecera IP en la operación.

Fig 4.13: Estructura de una cabecera ESP.

Uno de los parámetros adicionales que hacen interesante a ***ESP*** es la inclusión opcional de ***IVs.*** La finalidad de ***IV, Inicialization Vector*** consiste en hacer que el resultado cifrado de dos ***payloads*** idénticos resulte totalmente distinto. Esto impedirá romper el cifrado mediante análisis de frecuencia.

De igual modo, al utilizar ***ESP*** junto a ***SNAT*** existen problemas al no utilizar puertos. A la hora de tratar de ***de-NATear*** el paquete resultará imposible pues no existirá información de cómo hacerlo. En estos casos es necesario utilizar ***NAT-T*** o ***NAT Traversal*** que encapsulará ***ESP*** dentro de datagramas UDP para que ***SNAT*** funcione correctamente.

IKE, Internet Key Exchanging Protocol

Consiste en un protocolo ideado para la autenticación de las partes de la comunicación y el establecimiento de las claves simétricas. Para la autenticación es posible utilizar claves precompartidas, certificados o ***Kerberos.*** Una vez que la fase de autenticación se realiza, donde se ha establecido la ***ISAKMP SA***, es cuando ***IKE*** se encarga de crear las ***SA*** y por lo tanto actualizar las ***SAD*** implicadas. ***IKE*** por lo tanto se divide en dos fases, la fase 1 y la fase 2. En la primera de ellas se establece un canal de comunicación seguro utilizando ***Diffie-Hellman.*** De igual modo, en esta fase se realizará la autenticación de las partes implicadas en una comunicación utilizando clave precompartida, certificados o ***Kerberos*** entre otros mecanismos de autenticación. Posteriormente, en la fase 2 se utilizará el canal seguro establecido durante la fase 1 para generar las ***SA*** necesarias para establecer una comunicación segura.

IPsec con Linux

La pila ***IPsec*** se incluyó de forma oficial y como parte del kemel ***Linux*** con la salida de la rama estable 2.6, allá por el año 2002. Hasta entonces, el método para utilizar ***IPsec*** en ***Linux*** consistía en utilizar FreS/WAN que fue el proyecto que realizó la primera implementación con soporte completo y OpenSource.

En ***Linux*** es posible utilizar la implementación nativa utilizando las herramientas del paquete ***ipsec-tools***, que ayudarán a establecer y mantener las ***SAD*** y ***SPD.*** Por otra parte, para la gestión automática de claves y SA, es posible utilizar diferentes implementaciones de ***IKE*** como son ***racoon*** u ***OpenSwan.*** A continuación se verán una serie de ejemplos utilizando diferentes métodos de implementación de ***IPsec.***

Ejemplo IPsec "a mano"

En este caso tan solo es necesario instalar las herramientas necesarias para interactuar con la pila ***IPsec*** del núcleo. Basta con instalar el paquete ***ipsec-tools*** en el caso de Debian.

```
# apt-get install ipsec-tools
```

Antes de empezar con la configuración se ha de tener claro qué es lo que se desea hacer. En este caso, al tratarse de un ejemplo de inicio y básico lo que se pretende hacer es proteger el tráfico ICMP entre dos máquinas utilizando el modo transporte. Puede resultar algo sin mucho sentido, pero sirve para entender el comportamiento y funcionamiento de ***IPsec.*** Como es obvio, entrará en juego la configuración de dos máquinas, ambas en la misma red y sin ***NAT*** intermedio.

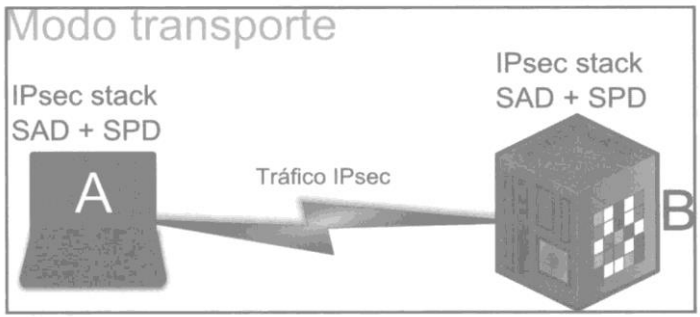


Fig 4.14: Protección del tráfico entre dos máquinas.

Es decir, se tiene que especificar en ambas máquinas cómo se va a proteger el tráfico entre ellas y para qué tipo de tráfico se aplicará dicha protección. Todo ello, con la ayuda del comando ***setkey,*** en cuyo manual se detallan todas sus opciones. Para este caso en concreto se utilizará un script ejecutable utilizando como intérprete ***setkey.*** La estructura del script podría ser la siguiente.

```
#!/usr/sbin/setkey -f

# Primero se establecerán las entradas para la SAD, Security Association
# Database. Es decir, el cómo se protegerá el tráfico entre las máquinas.

# A continuación se crearán las reglas o políticas para los diferentes tipos de
# tráfico y direcciones de la comunicación. SPD, Security Policy Database.
```

El modificador -/fdel comando ***setkey*** se utiliza para indicar que se analicen e interpreten las líneas que se establezcan a continuación, siendo posible utilizar comentarios con el símbolo almohadilla, #.

En el caso de Debían en lugar de Scripts se pueden introducir las líneas de ***setkey*** dentro del fichero / ***etc/ipsec-tools.conf,*** que será cargado de forma conveniente de acuerdo al script de inicio ***/etc/init.d/ setkey.*** Por defecto ese script no se ejecuta pues hay un parámetro para anular su ejecución en el fichero ***/etc/default/setkey.*** Tan solo hay que descomentar la línea ***RUN_SETKEY=yes.***

Configuración inicial y pruebas

Lo primero a realizar es crear un fichero de texto plano en cualquier ruta. En este caso en ***/usr/local/ sbin*** por estar contenida en el ***path*** del usuario root. Por supuesto, habrá que asignar permisos de ejecución para el usuario root al script. El contenido del fichero en la máquina A será el siguiente:

```
#!/usr/sbin/setkey -f

# Primero la definición de las SA, Security Associations
# ------------------------------------------------------
```

```
add 192.168.87.55 192.168.87.53 ah 33101 -A hmac-sha1 "Sup3rS3cr3t_P4ssW0rd";

# Definición de las SP, Security Policies
# ----------------------------------------
spdadd 192.168.87.55 192.168.87.53 icmp -P out ipsec
       ah/transport//require;
```

A continuación se explican las líneas utilizadas y cada uno de sus parámetros para este ejemplo. Todo está muy detallado en el ***man*** de ***setkey***, como viene siendo habitual en casi cualquier aplicación. Es muy importante recordar que por cada entrada que se desee añadir, esta debe terminar en punto y coma.

Con el modificador ***add*** se añade una entrada a la ***SAD.*** Sus parámetros por orden de aparición tienen el siguiente cometido.

192.168.87.55 y 192.168.87.53 son respectivamente las direcciones IP origen y destino de la asociación. Es decir, se está diciendo cómo proteger los paquetes que se indiquen en la ***SPD,*** cuando esas sean las direcciones implicadas y en ese orden.

- ***ah***, 33101 y ***-A hmac-shal***. Con ***AH*** se indica el tipo de protección para ese tipo de paquetes. El valor elegido al azar 33101 corresponde al ***SPI***, ***Security Parameter Index.*** Con ***-A*** se indica el tipo de algoritmo de autenticación para ***AH.*** En este caso se ha elegido ***hmac- shal*** seguido de una clave estática de 160 bits en este caso. La clave debe ir entre comillas dobles.

De igual modo, con el modificador ***spdadd*** se crea una nueva entrada en la ***SPD.*** Las direcciones IP vuelven a tener el mismo significado que en el caso de ***add.***

- ***icmp*** se ha especificado en esta ocasión para que solo se aplique ***IPsec*** a los paquetes de tipo ***icmp*** de las direcciones IP indicadas. Este valor podría haber sido tcp, udp o incluso ***any*** para englobar a todos los protocolos. El listado de los protocolos puede obtenerse del fichero ***/etc/protocols.***
- ***-P out ipsec*** indica la política en sí. Es decir, en este caso se especifica que esta política se aplica a los paquetes salientes. El valor puede ser ***in, out*** o ***fwd.*** Con respecto a la palabra ***ipsec*** se está indicando que para los paquetes con esas características se les aplique ***IPsec*** para protegerlos con la ***SA*** que corresponda.
- ***ah/trasnport//require*** indica el protocolo utilizado para ***IPsec***, su modo y el nivel de ***IPsec.*** En este ejemplo se indica que para el cualquier paquete ***ICMP*** saliente con dirección origen 192.168.87.55 y destino 192.168.87.53 se utilice ***AH*** en modo transporte requiriendo siempre la aplicación de ***IPsec.*** Se puede observar la falta de un parámetro entre las dos barras pero solo es necesario para indicar las direcciones origen y destino de los ***End-Points*** en modo túnel. Y ¿cómo se protegería ese tipo de paquete? - Se utilizarían las ***SA*** definidas anteriormente. Más adelante se explicará otro valor disponible para el nivel o ***level*** de ***IPsec.***

Una vez salvado el script anterior podría ejecutarse para aplicar tanto la política como la ***SA.*** Para consultar las ***SAD*** y ***SPD*** se puede utilizar el comando ***setkey*** con los parámetros ***-D*** y ***-DP*** respectivamente. La salida de ambos comandos es la siguiente.

```
root@test:~# setkey -D
192.168.87.55 192.168.87.51
        ah mode=transport spi=33101(0x0000814d) reqid=0(0x00000000)
        A: hmac-sha1  53377033 72533363 7233742e 50343473 57307264
        seq=0x00000000 replay=0 flags=0x00000000 state=mature
        created: Jan 31 21:46:19 2013   current: Jan 31 21:46:28 2013
        diff: 9(s)      hard: 0(s)      soft: 0(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=0 pid=2186 refcnt=0
root@test:~# setkey -DP
192.168.87.55[any] 192.168.87.51[any] icmp
        out prio def ipsec
        ah/transport//require
        created: Jan 31 21:46:19 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1 seq=0 pid=2187
        refcnt=1
```

Se observa que el kemel ya tiene cargadas las reglas introducidas por el script creado anteriormente. Ahora llega el momento de enviar un ICMP Echo request, ***ping*** a la máquina 192.168.87.51. ¿Qué ocurrirá?

```
root@test:~# ping -c1 192.168.87.53
PING 192.168.87.53 (192.168.87.53) 56(84) bytes of data.

--- 192.168.87.53 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Tal y como se aprecia en la salida anterior, no es posible obtener la respuesta a la solicitud ICMP. Es necesario visualizar en este caso qué está llegando a la máquina destino para entender lo que sucede. Se hace uso de ***Wireshark*** por ejemplo.

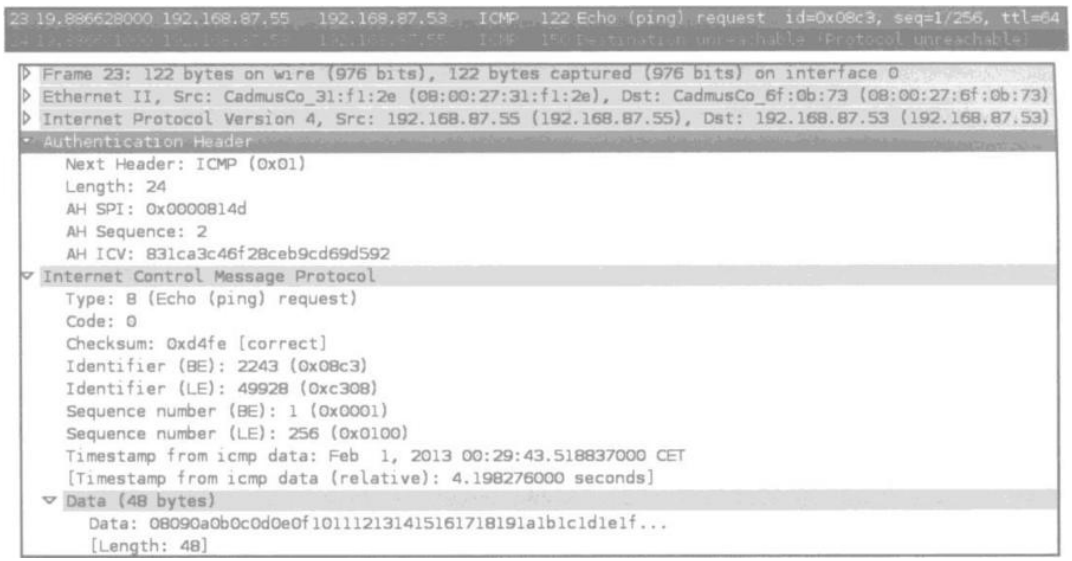


Fig 4.15: Salida de *Wireshark.*

El paquete llega a la máquina destino con la cabecera ***AH*** incluyendo los parámetros explicados anteriormente como son el ***SPI***, ***Next Header*** con valor ***icmp*** ya que es lo que se protege en este caso, etc. En el caso del ***SPI***, con valor ***0x0000814d*** equivalente a 33101 que fue el establecido para la única ***SA*** definida hasta el momento.

El problema está en que la máquina ***B***, con IP 192.168.87.53 no es capaz de resolver el protocolo ***IPsec*** asociado al paquete recibido pues no dispone de ninguna ***Security Association*** en su ***SAD.*** Al estar definiendo las ***SA*** de forma manual, sin utilizar ***IKE***, la máquina nunca sabrá interpretar el paquete. La solución consiste en agregar una entrada a la SA de la máquina ***B*** con el siguiente aspecto.

```
#!/usr/sbin/setkey -f

# Definición de la/s SA en la máquina B, 192.168.87.53
# ----------------------------------------------------
add 192.168.87.55 192.168.87.53 ah 33101 -A hmac-sha1 "Sup3rS3cr3t_P4ssW0rd";
```

Es decir, se indica cómo interpretar y procesar los paquetes ^4//desde la dirección IP 192.168.87.55. Una vez creada la ***SA*** se lanzaría el comando ***setkey -D*** en esta máquina ***B*** para comprobar el resultado.

```
root@wizi:~# setkey -D
192.168.87.55 192.168.87.53
        ah mode=transport spi=33101(0x0000814d) reqid=0(0x00000000)
```

```
A: hmac-sha1  73377033 72533363 7233742e 50343473 57307264
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Feb  1 20:29:18 2013current: Feb  1 20:29:22 2013
diff: 4(s)     hard: 0(s)     soft: 0(s)
last:                         hard: 0(s)     soft: 0(s)
current: 0(bytes)     hard: 0(bytes) soft: 0(bytes)
allocated: 0   hard: 0soft: 0
sadb_seq=0 pid=4111 refcnt=0
```

Repitiendo de nuevo la prueba de eco desde la máquina ***A*** se debería obtener un resultado positivo, ya que se ha agregado la ***SA*** necesaria.

```
root@test:~# ping -c1 192.168.87.53
PING 192.168.87.53 (192.168.87.53) 56(84) bytes of data.
64 bytes from 192.168.87.53: icmp_req=1 ttl=64 time=0.481 ms

--- 192.168.87.53 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.481/0.481/0.481/0.000 ms
```

La máquina ***B***, con dirección IP 192.168.87.53 ha sabido interpretar el datagrama protegido con ***AH*** y por lo tanto ha devuelto la respuesta a la solicitud ***ping*** enviada por ***A.*** Si se realiza una captura con ***Wireshark*** se puede observar como efectivamente la máquina ***B*** responde y además se aprecia una diferencia de longitud del paquete entre la solicitud de eco y la respuesta. La primera tiene una longitud de 122 bytes y sin embargo la respuesta de 98 bytes.

```
48 7.037649000  192.168.87.55   192.168.87.53   ICMP   122 Echo (ping) request  id=0x07a6, seq=1/256, ttl=64
49 7.037725000  192.168.87.53   192.168.87.55   ICMP    98 Echo (ping) reply    id=0x07a6, seq=1/256, ttl=64
```

Fig 4.16: Respuesta al tráfico ICMP protegido con AH.

La variación de longitud observada en la captura sucede porque en la petición se está agregando la cabecera ***AH*** con el ***SPI, HMAC*** y demás datos para la verificación posterior del datagrama. Todo ello suman 24 bytes, que unidos a los 98 bytes del datagrama original dan resultado a los 122 bytes. Observando este hecho y la configuración realizada en ambas máquinas se llega a la conclusión de que solo se ha configurado la protección de la comunicación en un sentido, de ***A*** hacia ***B.***

Protección de la comunicación en ambos sentidos

Debido a la configuración anterior, sería posible inyectar tráfico en la máquina ***A*** haciendo una suplantación de identidad a la máquina ***B.*** Para corregir esto es necesario configurar ambas máquinas para que protejan su tráfico entrante y saliente. Se continúa con el ejemplo de tráfico ***ICMP***, pero

todo lo mostrado a continuación es aplicable a cualquier protocolo que soporte ***IPsec*** entre los que se encuentran TCP y UDP.

Configuración de la máquina A
El script de la máquina con dirección IP 192.168.87.55 quedaría del siguiente modo.

```
#!/usr/sbin/setkey -f

# Máquina A
# Primero la definición de las SA, Security Associations
# ------------------------------------------------------
add 192.168.87.55 192.168.87.53 ah 33101 -A hmac-sha1 "Sup3rS3cr3t_P4ssW0rd";
add 192.168.87.53 192.168.87.55 ah 12345 -A hmac-md5 "Qw3rtyu10P4sdfgh";

# Definición de las SP, Security Policies
# ---------------------------------------
spdadd 192.168.87.55 192.168.87.53 icmp -P out ipsec
       ah/transport//require;
spdadd 192.168.87.53 192.168.87.55 icmp -P in ipsec
       ah/transport//require;
```

Como se puede observar se ha añadido una ***SA*** especificando el otro sentido de la comunicación. Es decir, los paquetes desde ***B*** hacia ***A.*** Igualmente se ha utilizado ***md5*** para ***AH*** en lugar de ***shal***, ya que no es necesario que la configuración de ambas ***SA*** sean idénticas. Por otra parte, atendiendo a las ***SP***, se ha agregado una nueva entrada a la ***SPD*** indicando que el tráfico ***ICMP*** entrante desde la máquina ***B*** debe ser protegido obligatoriamente por ***IPsec*** utilizando ***AH*** en modo transporte.

Configuración de la máquina B
En el caso de la configuración de ***B*** lo único que se modificará será las líneas de las políticas. Concretamente intercambiar ***in*** por ***out.***

```
#!/usr/sbin/setkey -f

# Máquina B
# Primero la definición de las SA, Security Associations
# ------------------------------------------------------
add 192.168.87.55 192.168.87.53 ah 33101 -A hmac-sha1 "Sup3rS3cr3t_P4ssW0rd";
add 192.168.87.53 192.168.87.55 ah 12345 -A hmac-md5 "Qw3rtyu10P4sdfgh";
```

```
# Definición de las SP, Security Policies
# ----------------------------------------
spdadd 192.168.87.55 192.168.87.53 icmp -P in ipsec
       ah/transport//require;
spdadd 192.168.87.53 192.168.87.55 icmp -P out ipsec
       ah/transport//require;
```

Se lanza de nuevo un ***ping*** desde ***A*** hacia ***B*** una vez actualizadas tanto la ***SAD*** y la ***SPD*** con la configuración anterior. Vuelve a funcionar sin problemas y en ***Wireshark*** esta vez se observa lo siguiente.

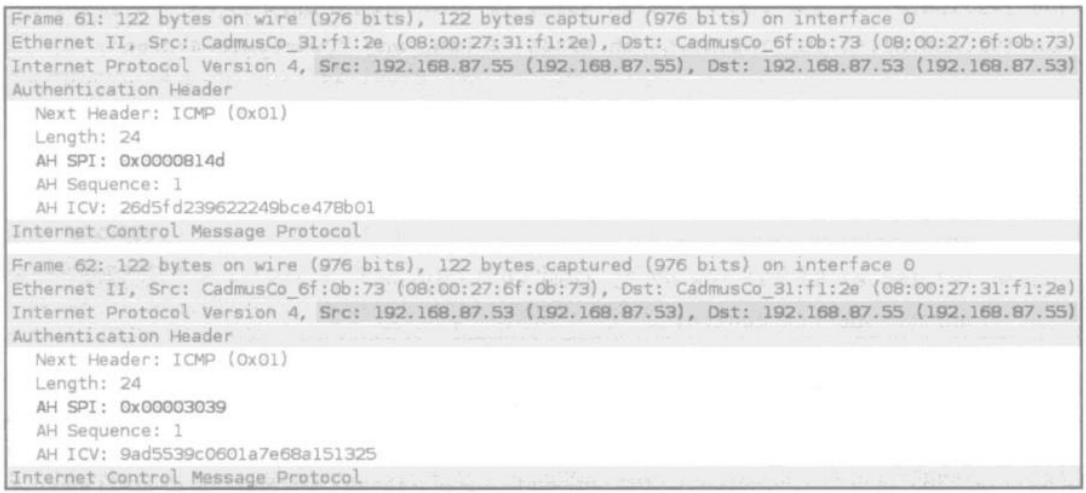


Fig 4.17: Cabeceras AH en ambos sentidos.

Tanto en la petición como en la respuesta se ha protegido el datagrama utilizando ***AH.*** Se puede destacar que en cada caso el ***SPI*** es diferente. Al disponer de varias entradas en la ***SAD***, se definieron valores diferentes para cada ***SA***, es decir un ***SPI*** diferente para cada entrada.

Los valores que se aprecian en la imagen son ***0x0000814d*** para la comunicación de ***A*** hacia ***B*** y ***0x00003039*** para la comunicación de ***B*** hacia ***A.*** Corresponden a los valores 33101 y 12345 respectivamente, definidos previamente en la configuración de las ***SA*** en ambas máquinas. Se concluye por lo tanto que en este instante la comunicación entrante y saliente de tipo ***ICMP*** entre ambas máquinas estará totalmente protegida.

Toda esta configuración funciona perfectamente y realiza su cometido de forma correcta, sin embargo resulta un poco incómodo tener que configurar las ***SA*** con las claves privadas, parámetros de los protocolos, etc. para cada una de las máquinas implicadas en una comunicación segura. Con el siguiente ejemplo ese paso quedará eliminado.

Ejemplos IPsec utilizando IKE

A continuación, siguiendo el mismo esquema anterior, se configurarán ambas máquinas para utilizar ***IKE*** y establecer de ese modo las ***SA*** de forma automática. Ello implica disponer de un demonio que implemente el protocolo ***IKE***, que en este caso será ***racoon.*** En primer lugar, en cada máquina que se vaya a utilizar ***IKE*** debe estar instalado ***racoon*** y por supuesto las ***ipsec-tools*** para disponer de ***setkey.*** En el caso de Debian la instalación de ***racoon*** tan solo precisa de la siguiente orden.

```
# apt-get install racoon
```

Con la instalación se ha creado el directorio de configuración de ***racoon*** con su fichero principal de configuración ***/etc/racoon/racoon.conf.*** Será el principal a modificar para poner en funcionamiento el entorno.

Como se comentaba con anterioridad, ***IKE*** ayudará a no tener que establecer las S A de forma manual e intercambiar dicha información entre las máquinas. Sin embargo sí es necesario configurar las ***SP*** en cada máquina. Cuando una ***SP*** requiera de una ***SA*** será cuando el demonio ***racoon*** en este caso, o el protocolo ***IKE***, entre enjuego para generar la SA necesaria.

Configuración mediante PSK

En el primer ejemplo se pasa a configurar ***racoon*** para la utilización de una clave precompartida entre dos máquinas para la autenticación en la fase 1 de ***IKE.*** Dicha clave deberá ser privada y estar protegida con los permisos adecuados en cada una de las máquinas configuradas.

Configuración de la máquina A

El fichero de configuración de la máquina con dirección **IP 192.168.87.55** tiene el siguiente aspecto.

```
path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode aggressive,main;

        my_identifier address;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 2 ;
        }
```

```
        generate_policy off;
}

sainfo anonymous
{
        pfs_group 1;
        encryption_algorithm 3des ;
        authentication_algorithm hmac_sha1;
                compression_algorithm deflate ;
}
```

La configuración se divide en dos partes, por un lado la correspondiente a la fase 1 en la que se intercambian las claves simétricas para establecer la comunicación segura utilizando ***Diffie-Hellman.*** Todos los parámetros necesarios para esa fase están contenidos dentro de la declaración ***remóte.*** Con respecto a la fase 2, cuyos parámetros están contenidos bajo la sección ***sainfo***, es donde se definen los parámetros propios de de los protocolos de ***IPsec*** como por ejemplo ***AH y ESP.***

Es posible definir secciones, tanto ***remóte*** como ***sainfo*** especificando hosts o subredes. Sin embargo, si se desea dotar de más flexibilidad la configuración o simplemente se están realizando pruebas con ***racoon***, es posible utilizar ***anonymous.*** A cualquier petición de cualquier cliente que no esté definido con una sección propia se aplicarán los parámetros de las secciones establecidas como ***anonymous.*** Esto último también resulta útil cuando se utiliza ***IPsec*** para VPN de tipo ***Road-Warrior*** y no es posible saber la IP de la máquina solicitante.

Racoon dispone de una gran variedad de parámetros para su configuración en diferentes situaciones, siendo la configuración anterior básica para no complicar en exceso el uso de ***IKE.*** A continuación se exponen los parámetros utilizados divididos por las dos fases ya mencionadas.

	Parámetro	Explicación
Fase 1 **Sección remote**	**exchange_mode**	Es el modo en que se realiza el intercambio de información durante la fase 1. Sus valores pueden ser *main* y/o *aggressive*. La diferencia entre ambos es el número de pasos utilizados en la comunicación. Siendo *aggressive* menos seguro pero más rápido a la hora del intercambio de información. El hash basado en la PSK será transmitido sin cifrar y podría ser capturado por un atacante. A pesar de ello es necesario utilizar el modo *aggressive* en determinadas configuraciones.

	Parámetro	Explicación
Fase 1 Sección remote	my_identifier	Especifica cómo se identificará la máquina frente al otro host. En este caso se ha usado el parámetro *address* para utilizar la dirección IP. En caso de utilizar un certificado sería necesario configurar *asn1dn* para utilizar le *DN*, *Distingished Name*.
	proposal	Se definen los parámetros con los que se tratará de establecer la comunicación segura en la fase 1. Tanto los parámetros de cifrado como los autenticación. De igual modo se define el modo de autenticación de entre los que dispone *racoon*. Es posible seleccionar entre *psk*, *Kerberos*, clave pública, etc.
	generate_policy	Es posible indicar a *racoon* que genere las políticas necesarias para las *SA* en caso de no existir. Es preferible dejar este parámetro a *off* para evitar conflictos con otras *SP* que pudieran estar establecidas.
Fase 2 Sección sainfo	pfs_group	Se trata de otro parámetro para tratar de securizar aún más el intercambio de claves con *Diffie-Hellman*, esta vez durante la fase 2.
	encryption_algorithm	Aquí se declaran los posibles algoritmos de cifrado al crear una *SA* donde *ESP* sea necesario.
	authentication_algorithm	El mismo cometido que el caso anterior pero esta vez con *AH*. Al igual que con *ESP*, pueden consultarse los diferentes algoritmos en el *man* de *setkey* o *racoon.conf*.
	compression_algorithm	Se establece a *deflate* y es el mecanismo de compresión cuando se utiliza *IPComp* o compresión para *IPsec*.

Lo anterior tan solo es una explicación básica de los parámetros utilizados en esta configuración. Sin embargo, todo está muy detallado en el ***man*** de ***racoon.conf.***

El parámetro inicial del fichero claramente indica la ruta del fichero donde se almacenan las ***psk*** para los diferentes extremos de la comunicación. El fichero tiene una estructura de tipo host ***psk*** por cada línea. En este caso el fichero ***/etc/racoon/psk.txt*** tendrá un aspecto como el siguiente.

```
192.168.87.53   S3cr3tP4ssW0rD
```

A partir de este momento tan solo hay que reiniciar o recargar el servicio ***racoon*** para aplicar la nueva configuración, además de aplicar las ***SP*** necesarias en cada caso. Para este ejemplo, el script de ***IPsec*** se ha modificado.

```
#!/usr/sbin/setkey -f

# Definición de las SP, Security Policies
# ----------------------------------------
spdadd 192.168.87.55 192.168.87.53 icmp -P out ipsec
       ah/transport//require;
spdadd 192.168.87.53 192.168.87.55 icmp -P in ipsec
       ah/transport//require;
```

Es decir, ahora solo se están aplicando nuevas ***SP*** y no ***SA*** que será cosa de ***racoon.*** Se puede verificar la configuración realizando un volcado de la ***SAD*** y ***SPD*** con ***setkey.***

```
root@test:~# setkey -D
No SAD entries.
root@test:~# setkey -DP
192.168.87.53[any] 192.168.87.55[any] icmp
        fwd prio def ipsec
        ah/transport//use
        created: Feb  3 13:08:28 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=82 seq=1 pid=1892
        refcnt=1
192.168.87.53[any] 192.168.87.55[any] icmp
        in prio def ipsec
        ah/transport//use
        created: Feb  3 13:08:28 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=72 seq=2 pid=1892
        refcnt=1
192.168.87.55[any] 192.168.87.53[any] icmp
        out prio def ipsec
```

```
        ah/transport//use
        created: Feb  3 13:08:28 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=65 seq=3 pid=1892
        refcnt=1
```

Configuración de la máquina B

El fichero de configuración de la máquina con dirección **IP 192.168.87.53** tiene el mismo aspecto que el de la máquina

```
path pre_shared_key "/etc/racoon/psk.txt";

remote anonymous
{
        exchange_mode aggressive,main;

        my_identifier address;

        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group 2 ;
        }
        generate_policy off;
}

sainfo anonymous
{
        pfs_group 1;
        encryption_algorithm 3des ;
        authentication_algorithm hmac_sha1;
                compression_algorithm deflate ;
}
```

Su fichero ***/etc/racoon/psk.txt*** es igual salvo por la dirección **IP,** que es la de la máquina opuesta al igual que ocurría en la máquina ***A.***

```
192.168.87.55   S3cr3tP4ssW0rD
```

Realmente lo que se está indicando es que cuando la máquina se identifique con su dirección IP se utilice esa línea como ***psk.***

Al igual que en el caso anterior es necesario actualizar la ***SPD*** con la configuración deseada y dejar el que la ***SAD*** sea actualizada por ***raccon.*** Tras recargar el ***racoon*** la configuración de esta máquina sería la siguiente.

```
root@wizi:~# setkey -D
No SAD entries.
root@wizi:~# setkey -DP
192.168.87.53 192.168.87.55 icmp
        out prio def ipsec
        ah/transport//require
        created: Feb  3 13:17:18 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=177 seq=1 pid=4092
        refcnt=1
192.168.87.55 192.168.87.53 icmp
        fwd prio def ipsec
        ah/transport//require
        created: Feb  3 13:17:18 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=170 seq=2 pid=4092
        refcnt=1
192.168.87.55 192.168.87.53 icmp
        in prio def ipsec
        ah/transport//require
        created: Feb  3 13:17:18 2013  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=160 seq=3 pid=4092
        refcnt=1
```

Probando la configuración

Ahora que todo está correctamente configurado para proteger las comunicaciones ***icmp*** entre ambas máquinas se realizará un prueba con los siguientes pasos para comprobar el correcto funcionamiento del entorno.

1. Realizar ***ping*** desde ***A*** hacia ***B.***
2. Visualización del ***syslog*** de ***B*** para verificar que ***IKE*** está funcionando.
3. Verificar las ***SAD*** de ambas máquinas para ver qué ha sucedido.
4. Captura de red con ***Wireshark*** en la máquina ***B***, verificando el funcionamiento de ***IKE*** e ***IPsec.***

Para el***ping*** de la máquina se utilizarán cuatro peticiones en lugar de una, debido a que probablemente los primeros paquetes se pierdan mientras ***racoon*** está realizando la configuración de las ***SAD.***

```
root@test:~# ping -c4 192.168.87.53
PING 192.168.87.53 (192.168.87.53) 56(84) bytes of data.
64 bytes from 192.168.87.53: icmp_req=3 ttl=64 time=0.461 ms
64 bytes from 192.168.87.53: icmp_req=4 ttl=64 time=0.397 ms

--- 192.168.87.53 ping statistics ---
4 packets transmitted, 2 received, 50% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.397/0.429/0.461/0.032 ms
```

En la máquina ***B*** el ***syslog*** tendrá un aspecto como el siguiente, diferenciándose las fases 1 y 2 del protocolo ***IKE.*** Puede observarse que el modo para la primera fase ha sido ***Aggressive***, utilizando ***psk*** y que posteriormente se han generado las ***SA*** necesarias.

```
Feb  3 13:37:26 wizi racoon: INFO: respond new phase 1 negotiation: 192.168.87
.53[500]<=>192.168.87.55[500]
Feb  3 13:37:26 wizi racoon: INFO: begin Aggressive mode.
Feb  3 13:37:26 wizi racoon: INFO: received Vendor ID: DPD
Feb  3 13:37:26 wizi racoon: [192.168.87.55] NOTIFY: couldn't find the proper
pskey, try to get one by the peer's address.
Feb  3 13:37:26 wizi racoon: INFO: ISAKMP-SA established 192.168.87.53[500]-
192.168.87.55[500] spi:65aa7107fe603622:9652a18ef917fda3
Feb  3 13:37:26 wizi racoon: [192.168.87.55] INFO: received INITIAL-CONTACT
Feb  3 13:37:27 wizi racoon: INFO: respond new phase 2 negotiation: 192.168.87
.53[500]<=>192.168.87.55[500]
Feb  3 13:37:27 wizi racoon: INFO: IPsec-SA established: AH/Transport
192.168.87.53[500]->192.168.87.55[500] spi=89670070(0x55841b6)
Feb  3 13:37:27 wizi racoon: INFO: IPsec-SA established: AH/Transport
192.168.87.53[500]->192.168.87.55[500] spi=206557250(0xc4fd042)
```

En ambas máquinas se habrán actualizado las ***SAD*** con la información gracias a ***IKE*.** Las siguientes líneas corresponden a las ***SAD*** de las máquinas ***A y B*** respectivamente.

```
root@test:~# setkey -D
192.168.87.55 192.168.87.53
        ah mode=transport spi=89670070(0x055841b6) reqid=0(0x00000000)
        A: hmac-sha1  98ef2d43 a51e8d4d f822dd79 c1140c58 ad6c3103
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Feb  3 13:37:27 2013   current: Feb  3 13:52:16 2013
        diff: 889(s)    hard: 28800(s)  soft: 23040(s)
        last: Feb  3 13:37:28 2013      hard: 0(s)      soft: 0(s)
        current: 128(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 2    hard: 0 soft: 0
        sadb_seq=1 pid=1960 refcnt=0
192.168.87.53 192.168.87.55
        ah mode=transport spi=206557250(0x0c4fd042) reqid=0(0x00000000)
        A: hmac-sha1  1279397b ef15411a cae14a1f addcf6d5 a9e57788
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Feb  3 13:37:27 2013   current: Feb  3 13:52:16 2013
        diff: 889(s)    hard: 28800(s)  soft: 23040(s)
        last: Feb  3 13:37:28 2013      hard: 0(s)      soft: 0(s)
        current: 128(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 2    hard: 0 soft: 0
        sadb_seq=0 pid=1960 refcnt=0
# ---------
root@wizi:~# setkey -D
192.168.87.53 192.168.87.55
       ah mode=transport spi=206557250(0x0c4fd042) reqid=0(0x00000000)
       A: hmac-sha1  1279397b ef15411a cae14a1f addcf6d5 a9e57788
       seq=0x00000000 replay=4 flags=0x00000000 state=mature
       created: Feb  3 13:37:27 2013current: Feb  3 13:52:08 2013
       diff: 881(s)   hard: 28800(s) soft: 23040(s)
       last: Feb  3 13:37:28 2013   hard: 0(s)     soft: 0(s)
       current: 128(bytes)   hard: 0(bytes) soft: 0(bytes)
       allocated: 2   hard: 0soft: 0
       sadb_seq=1 pid=4497 refcnt=0
```

```
192.168.87.55 192.168.87.53
        ah mode=transport spi=89670070(0x055841b6) reqid=0(0x00000000)
        A: hmac-sha1  98ef2d43 a51e8d4d f822dd79 c1140c58 ad6c3103
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Feb  3 13:37:27 2013current: Feb  3 13:52:08 2013
        diff: 881(s)   hard: 28800(s) soft: 23040(s)
        last: Feb  3 13:37:28 2013    hard: 0(s)     soft: 0(s)
        current: 128(bytes)   hard: 0(bytes) soft: 0(bytes)
        allocated: 2   hard: 0 soft: 0
        sadb_seq=0 pid=4497 refcnt=0
```

Si se presta la suficiente atención sobre el log anterior y las ***SAD*** de ambas máquinas, puede observarse que las ***SPI*** de las ***SAD*** resultan ser las mismas que aparecen en el log. Parece que todo ha funcionado perfectamente.

En ***Wireshark*** puede observarse todo el proceso con la siguiente secuencia de datagramas. Se observa como inicialmente la máquina ***A*** realiza la solicitud de eco y como posteriormente comenzar a funcionar IKE para la generación de las ***SA.***

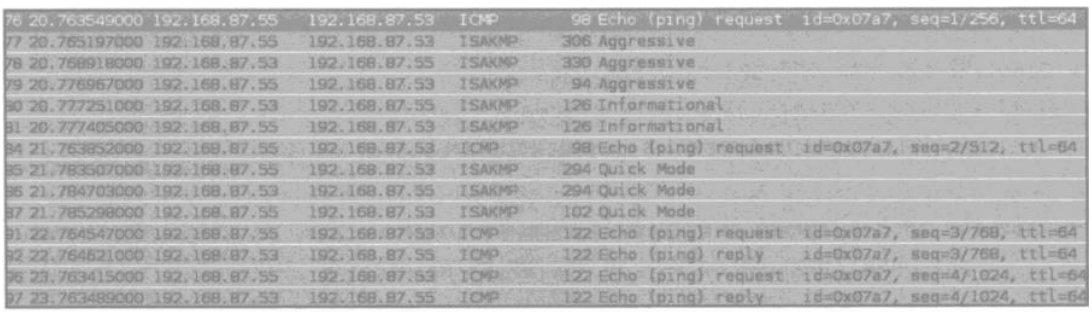


Fig 4.18: IPsec e IKE.

Configuración en modo túnel

Hasta ahora solo se habían creado configuraciones para uno de los modos en los que puede trabajar ***IPsec.*** A continuación se configura un entorno en modo túnel utilizando en este caso tres máquinas para ello.

La idea es que una máquina se comunique con otra en una red diferente utilizando para ello un router intermedio. El tráfico entre la máquina que inicia la comunicación y el router estará protegido por ***IPsec***, siendo innecesaria la configuración en la tercera máquina.

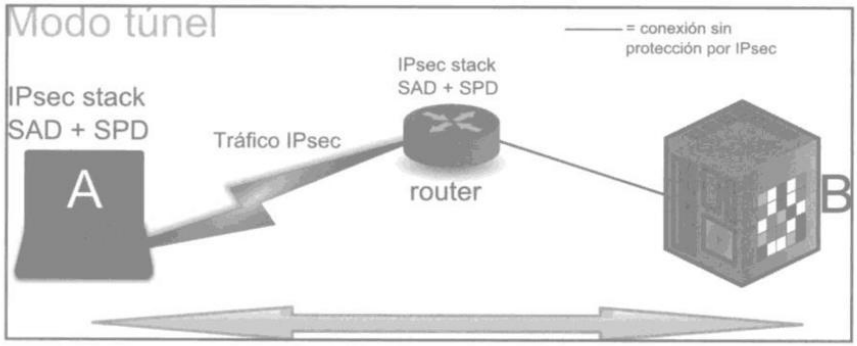


Fig 4.19: IPsec en modo túnel.

La comunicación será iniciada por la máquina ***A*** con dirección IP 192.168.87.53, tratando de hacer ***ping*** a la máquina ***B*** con dirección IP 172.16.0.52.

Configuración de la máquina A

En primer lugar se configura ***racoon*** de forma adecuada. Siendo la configuración utilizada en este caso la idéntica a los ejemplos anteriores en modo transporte por lo que se obviará mostrar nuevamente dicha configuración. Para el caso de las ***SPD*** es necesario que el script quede del siguiente modo.

```
#!/usr/sbin/setkey -f

# Definición de las SP, Security Policies
spdadd 192.168.87.0/24 172.16.0.0/24 icmp -P out ipsec
       ah/tunnel/192.168.87.53-192.168.87.55/require;

spdadd 172.16.0.0/24 192.168.87.0/24 icmp -P in ipsec
        ah/tunnel/192.168.87.55-192.168.87.53/require;
```

Puede observarse que ahora el modo se ha definido como ***tunnel*** y adicionalmente, siendo totalmente necesario, se han definido los dos extremos del túnel separados por un guión.

Lógicamente es necesario configurar la tabla de rutas de la máquina ***A,*** ya que en este ejemplo el router que conecta las redes de ***A*** y ***B*** no es la puerta de enlace por defecto de ***A.*** Para ello se agrega la ruta de forma manual.

```
route add -net 172.16.0.0/24 gw 192.168.87.55
```

Configuración del router

En el router habrá que agregar las políticas correspondientes a la ***SPD*** para poder establecer la comunicación segura con la máquina A.

```
#!/usr/sbin/setkey -f
# Definición de las SP, Security Policies

spdadd 192.168.87.0/24 172.16.0.0/24 icmp -P in ipsec
        ah/tunnel/192.168.87.53-192.168.87.55/require;

spdadd 172.16.0.0/24 192.168.87.0/24 icmp -P out ipsec
        ah/tunnel/192.168.87.55-192.168.87.53/require;
```

Por supuesto, esta máquina está configurada como un router y posee dos interfaces de red.

```
root@test:~# cat /proc/sys/net/ipv4/ip_forward
1
root@test:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:31:f1:2e
          inet addr:192.168.87.55  Bcast:192.168.87.255  Mask:255.255.255.0
          ...

eth1      Link encap:Ethernet  HWaddr 08:00:27:d1:58:d2
          inet addr:172.16.0.1  Bcast:172.16.0.255  Mask:255.255.255.0
          ...
```

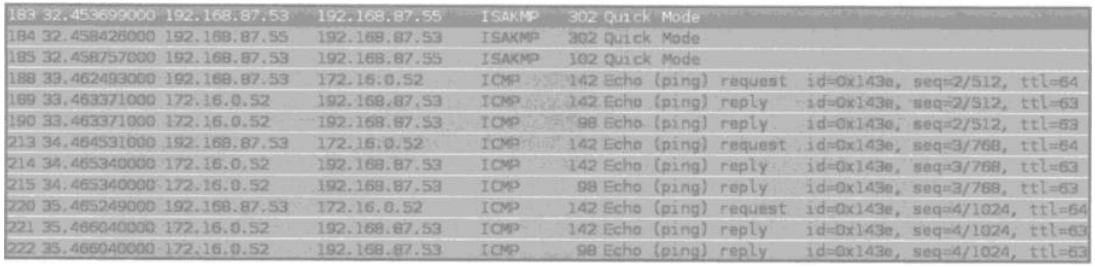
Probando la configuración

En la máquina ***B*** no ha sido necesaria ninguna configuración dado que no será partícipe de la comunicación ***IPsec*** y por otra parte tiene establecida como puerta de enlace por defecto la dirección del router 172.16.0.1.

El comando***ping*** ejecutado desde la máquina^ tendría un aspecto como el siguiente.

```
root@wizi:~# ping -c4 172.16.0.52
PING 172.16.0.52 (172.16.0.52) 56(84) bytes of data.
64 bytes from 172.16.0.52: icmp_req=2 ttl=63 time=1.03 ms
64 bytes from 172.16.0.52: icmp_req=3 ttl=63 time=0.854 ms
64 bytes from 172.16.0.52: icmp_req=4 ttl=63 time=0.862 ms
--- 172.16.0.52 ping statistics ---
4 packets transmitted, 3 received, 25% packet loss, time 3012ms
rtt min/avg/max/mdev = 0.854/0.918/1.038/0.084 ms
```

Por otra parte, la captura de datos realizada con ***Wireshark*** mostraría el siguiente aspecto.



183 32.453699000	192.168.87.53	192.168.87.55	ISAKMP	302 Quick Mode	
184 32.458428000	192.168.87.55	192.168.87.53	ISAKMP	302 Quick Mode	
185 32.458757000	192.168.87.53	192.168.87.55	ISAKMP	102 Quick Mode	
188 33.462493000	192.168.87.53	172.16.0.52	ICMP	142 Echo (ping) request	id=0x143e, seq=2/512, ttl=64
189 33.463371000	172.16.0.52	192.168.87.53	ICMP	142 Echo (ping) reply	id=0x143e, seq=2/512, ttl=63
190 33.463371000	172.16.0.52	192.168.87.53	ICMP	98 Echo (ping) reply	id=0x143e, seq=2/512, ttl=63
213 34.464531000	192.168.87.53	172.16.0.52	ICMP	142 Echo (ping) request	id=0x143e, seq=3/768, ttl=64
214 34.465340000	172.16.0.52	192.168.87.53	ICMP	142 Echo (ping) reply	id=0x143e, seq=3/768, ttl=63
215 34.465340000	172.16.0.52	192.168.87.53	ICMP	98 Echo (ping) reply	id=0x143e, seq=3/768, ttl=63
220 35.465249000	192.168.87.53	172.16.0.52	ICMP	142 Echo (ping) request	id=0x143e, seq=4/1024, ttl=64
221 35.466040000	172.16.0.52	192.168.87.53	ICMP	142 Echo (ping) reply	id=0x143e, seq=4/1024, ttl=63
222 35.466040000	172.16.0.52	192.168.87.53	ICMP	98 Echo (ping) reply	id=0x143e, seq=4/1024, ttl=63

Fig 4.20: Captura de datos con *Wireshark* en modo túnel (1).

Todo funciona correctamente habiéndose definido de forma automática en la máquina^ las siguientes entradas en su ***SAD.***

```
root@wizi:~ # setkey -D
192.168.87.53 192.168.87.55
        ah mode=tunnel spi=251427695(0x0efc7b6f) reqid=0(0x00000000)
        A: hmac-sha1  8c5f60dc c5d825b1 1a0f822b 5ed3669f 91fb7784
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Feb  3 16:29:26 2013current: Feb  3 16:40:02 2013
        diff: 636(s)   hard: 28800(s) soft: 23040(s)
        last: Feb  3 16:29:27 2013    hard: 0(s)     soft: 0(s)
        current: 252(bytes)   hard: 0(bytes) soft: 0(bytes)
        allocated: 3   hard: 0soft: 0
        sadb_seq=1 pid=5243 refcnt=0
192.168.87.55 192.168.87.53
        ah mode=tunnel spi=57876359(0x03731f87) reqid=0(0x00000000)
        A: hmac-sha1  80b9981d b5b1aee4 803207a1 31465d49 9433458c
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Feb  3 16:29:26 2013current: Feb  3 16:40:02 2013
        diff: 636(s)   hard: 28800(s) soft: 23040(s)
        last: Feb  3 16:29:27 2013    hard: 0(s)     soft: 0(s)
        current: 252(bytes)   hard: 0(bytes) soft: 0(bytes)
        allocated: 3   hard: 0soft: 0
        sadb_seq=0 pid=5243 refcnt=0
```

Al haber utilizado modo transporte, en la siguiente captura de datagramas con ***Wireshark*** queda de manifiesto el hecho la creación de una segunda cabecera IP. La nueva cabecera contiene la

información de los extremos del túnel, en cambio la cabecera inicial es la que contiene el inicio y el final de la comunicación.

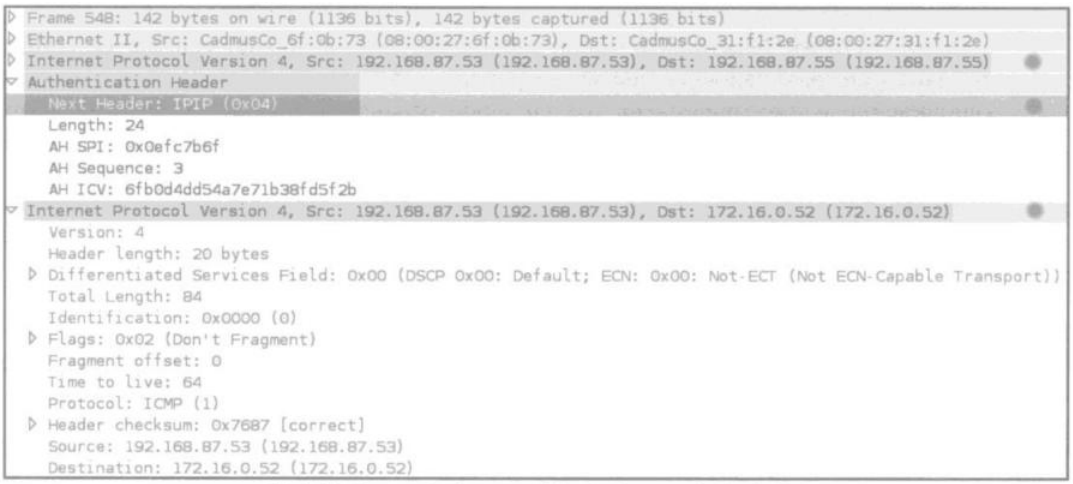


Fig 4.21: Captura de datos con *Wireshark* en modo túnel (2).

Todos los ejemplos mostrados anteriormente han sido sencillos para que la primera toma de contacto no conllevase demasiada dificultad. Obviamente esos ejemplos pueden ser modificados para utilizar ***ESP*** modificando tan solo las políticas en el caso de ***IKE***, autenticación mediante certificados utilizando ***racoon***, etc. Por otra parte, también existen opciones en ***racoon*** para actualizar de forma periódica las ***SA*** de una comunicación ***IPsec*** para aumentar la seguridad del entorno. Esto último se realiza con los parámetros ***lifetime***, los cuales están documentados en el ***man*** del fichero ***racoon. conf.*** En definitiva, dependiendo de las necesidades es posible configurar ***IPsec*** de muchos modos diferentes.

Por otra parte no resulta obligatorio utilizar ***racoon*** para implementar ***IPsec*** con ***IKE.*** Es posible utilizar ***OpenSwan*** o ***SírongSwan*** entre otras implementaciones, sin olvidar la combinación de ***IPsec*** con ***NAT-T***y l***2tp*** para crear VPN de tipo ***Road-Warrior***, etc.

4. IDS Snort

Como última parte de este capítulo se introduce el concepto de ***IDS.*** Un ***IDS***, ***Intrusión Detection System***, no es más que algún tipo de software capaz de detectar accesos no autorizados o fraudulentos a una máquina o una red. Un aspecto muy importante de este tipo de software son las bases de datos de reglas, actualizadas de forma frecuente por la comunidad o grupos especializados para dotar a los IDS de más potencia a la hora de detectar ataques. En este aspecto resulta muy parecido a un antivirus. Los IDS se categorizan en dos tipos según su modo de funcionamiento.

- ***HIDS***, ***Host Intrusión Detection System.*** Está ideado para ser instalado en cada máquina, y su principal cometido es verificar qué cambios se han realizado en dicha máquina tras un acceso indebido.
- ***NIDS***, ***NetWork Intrusión Detection System.*** Su funcionamiento consiste en realizar un análisis de los datagramas que viajan por una red de datos. Para ello es necesaria la configuración de una interfaz de red en modo promiscuo, aunque dependiendo de la topología de cada red y los intereses de los administradores de sistemas, habrá que elegir una zona idónea donde poner a esnifar esa interfaz. En principio, un software NIDS solo está preparado para alertar en caso de detección de intentos de ataque, paquetes mal formados, etc.; aunque es posible implementarlos junto a***firewalls*** para crear reglas de forma dinámica y mitigar rápidamente los ataques que pudieran detectarse.

En este capítulo se usará un ***NIDS*** por el hecho de estar tratando la protección de la red interna. Realmente podría haberse hablado de un ***NIDS*** en la protección perimetral, pues depende de dónde se configure se estará protegiendo una zona u otra.

Relacionado con las zonas y el análisis de paquetes de las mismas, se podrá configurar un ***NIDS*** de diferentes modos dependiendo de la electrónica de red, de la configuración de los cortafuegos, etcétera.

- Bien esnifando paquetes directamente desde un adaptador conectado a la boca de un ***switch.*** En este caso el único tráfico que se estaría analizando sería el de ***Broadcast*** de una red, intentos de escaneos a un segmento completo, etc. Probablemente se pierda mucha información configurando el ***NIDS*** de este modo.
- Utilizar dispositivos ***tap*** o ***switches*** gestionables para obtener más información. En este caso, haciendo uso por ejemplo de un ***switch*** gestionable, podría configurarse un ***port mirroring*** de una boca de ***switch*** que conecta con un ***firewall***, directamente con la conexión a Internet, etc.

En este caso se ha configurado el NIDS tal y como se muestra en la imagen 4.22 de la siguiente página.

Es decir, se está haciendo un ***port mirroring*** de la boca que lleva el tráfico de una red interna hacia la máquina que actúa de cortafuegos, enrutador y dispositivo de NAT. De igual modo se podría haber conectado el ***IDS*** de modo que se escanease el tráfico de la interfaz externa del ***router***, o lo que sería lo mismo, escanear el tráfico asociado a la dirección IP externa o pública de la red.

Sea como fuere la configuración escogida o necesaria, el software elegido para implementar el ***NIDS*** ha sido ***Snort***, ya que se trata de un proyecto de software libre muy conocido, sólido y altamente utilizado por la comunidad de usuarios. En este ejemplo se instalará ***Snort*** junto a una interfaz web que ayudará a visualizar las estadísticas que el ***NIDS*** va recopilando. Dicha interfaz se es conocida como ***BASE***, ***Basic Analysis and Security Engine.***

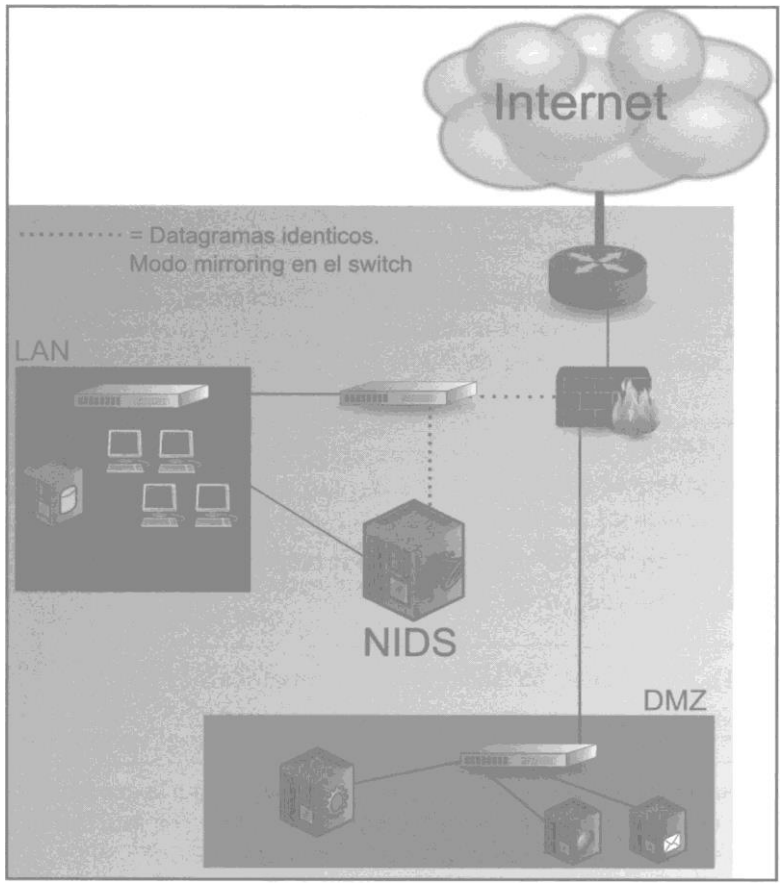


Fig 4.22: Configuración de un NIDS.

Instalación de Snort desde los repositorios oficiales de Debían

En este caso se ha optado por instalar ***Snort*** en este modo debido a la relativa facilidad de instalación de todos los componentes, lo que ayudará a tener un primera toma de contacto y una curva de aprendizaje menos pronunciada. Como se comentaba con anterioridad, se instalará de forma adicional una interfaz web para visualizar los resultados, por lo que será necesario instalar un sistema ***LAMP***, ***Linux, Apache, MySQL y PHP.*** De igual modo se instalará ***Snort*** de modo que guarde sus registros en una base de datos ***SQL*** utilizando ***MySQL***.

Instalación del entorno LAMP

Instalar este tipo de entornos resulta muy sencillo en cualquier distribución ***Linux*** actualmente. En Debian, como es lógico se instalará todo utilizando el gestor de paquetes ***apt.***

```
root@test:~# apt-get install apache2 php5 mysql-server
```

Posteriormente será necesario reconfigurar el paquete del servicio ***mysql*** para configurar la cuenta de administrador de la base de datos. Con el siguiente comando aparecerá un asistente para hacerlo fácilmente.

```
dpkg-reconfigure mysql-server-5.1
```

A pesar de que posteriormente se tratará la securización de entornos ***LAMP***, no es muy buena idea configurar ***Snort*** ni ***BASE*** utilizando el usuario ***root*** de ***MySQL.*** Es por ello que debe crearse un usuario para ***Snort*** con su propia base de datos. Los comandos a ejecutar en ***MySQL*** serían los siguientes.

```
mysql> create database snort;
Query OK, 1 row affected (0.00 sec)

mysql> grant usage on *.* to snort@localhost identified by '123abc.';
Query OK, 0 rows affected (0.00 sec)

mysql> grant all privileges on snort.* to snort@localhost;
Query OK, 0 rows affected (0.00 sec)
```

En este caso el usuario es ***snort***, con una base de datos llamada ***snort*** a la que tiene acceso con todos los privilegios. Por otra parte, el ***password*** establecido es ***123abc..*** Con esta última parte ya estaría completa la preparación del sistema para poder empezar a utilizar ***Snort.***

Instalación de Snort y BASE

En Debian el paquete con el que se instalará la versión de ***Snort*** para usar ***MySQL*** como ***back-end*** de almacenamiento de registros es ***snort-mysql.***

```
root@test:~# apt-get install snort-mysql
```

Junto a ese paquete se instalarán algunas dependencias como ***libpcap*** y algunos módulos de Perl entre otros. Los módulos de Perl se instalan para utilizar la herramienta ***oinkmaster***, cuyo objetivo es actualizar las reglas de ***Snort.***

Al iniciarse la configuración de ***Snort*** durante el proceso de instalación, se preguntará por el segmento de red que se desea analizar mediante la interfaz de red en modo promiscuo. Por otra parte en el último paso de instalación se indicará el modo para inicializar la base de datos de ***Snort***, ya que no

se realiza automáticamente. Según lo indicado en el último paso de la instalación hay que ejecutar la siguiente orden desde la terminal.

```
# zcat /usr/share/doc/snort-mysql/create_mysql.gz | mysql -u snort \
-D snort -p123abc.
```

Los parámetros del comando ***mysql*** corresponden al usuario, base de datos y contraseña, que fúeron creados anteriormente.

El siguiente paso consiste en instalar ***BASE***, el visualizador web de las estadísticas de ***Snort.***

```
root@test:~# apt-get install acidbase
```

Una vez que se pregunte por el tipo de base de datos a utilizar se debe seleccionar ***MySQL*** para este ejemplo e introducir los valores para el usuario ***snort*** creado anteriormente. La instalación finalizará sin más problemas y tan solo quedará un paso para dejar totalmente configurado ***BASE.*** Consiste en habilitar el acceso a su interfaz web a más direcciones o subredes. En concreto es necesario editar el fichero ***/etc/base/apache.conf*** para modificar o agregar la línea con la directiva ***allow.*** En este ejemplo se permitirán los acceso de las subredes 172.16.0.0/24 y 192.168.87.0/24, por lo que se agregarán al fichero las siguientes líneas.

```
allow from 127.0.0.0/255.0.0.0      # Esta entrada ya existía
allow from 172.16.0.0/255.255.255.0
allow from 192.168.87.0/255.255.255.0
```

Para finalizar el proceso de instalación, resulta necesario reconfigurar ***Snort*** para añadir parámetros adicionales tales como la interfaz de red utilizada en modo promiscuo que esnifará tráfico, reconfiguración de la base de datos, envío por correo electrónico, etc. Para ello se deben realizar se los siguientes pasos, estando indicado el primero de ellos en el fichero ***/usr/share/doc/snort-mysql/README-database.Debian.***

```
# rm /etc/snort/db-pending-config
# dpkg --configure --pending
# dpkg-reconfigure snort-mysql
```

El primero de los tres comandos elimina un fichero temporal generado por la instalación inicial de ***Snort.*** El segundo indica que se finalice la configuración tras la instalación inicial de ***Snort*** y por lo tanto el paquete quede completamente instalado de cara al sistema. Para finalizar se reconfigura el paquete como se indicaba anteriormente.

Las siguientes son algunas de las pantallas correspondientes al proceso de reconfiguración.

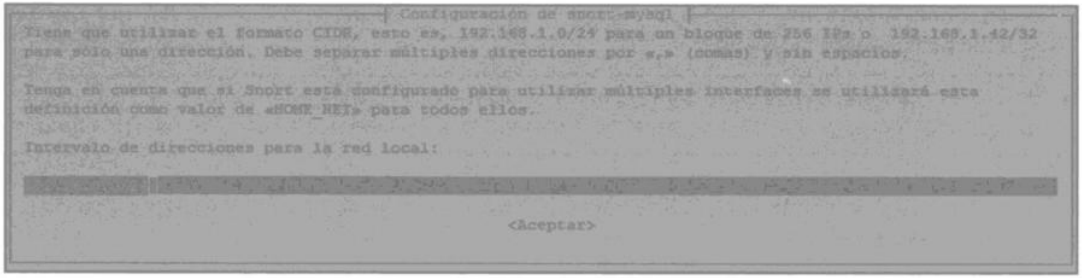


Fig 4.23: Configuración del segmento de red a monitorizar.

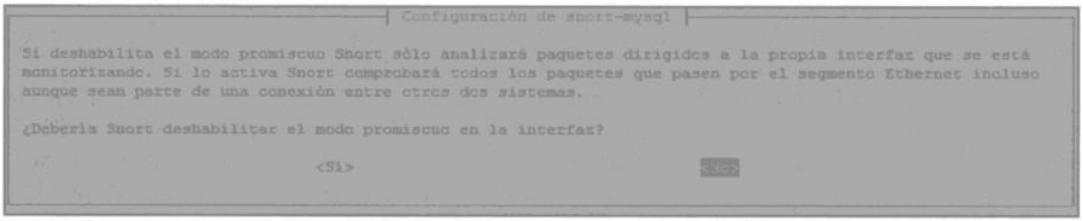


Fig 4.24: Habilitar el modo promiscuo de la interfaz elegida.

Snort en funcionamiento

Antes de continuar es necesario comprobar si ***Snort*** está capturando paquetes en la interfaz de red configurada anteriormente. Para ello se lanza ***Snort*** desde la terminal con la línea ***snort -i eth2*** -v. En este caso se puede observar como efectivamente se han capturado paquetes por lo que ***Snort*** está funcionando correctamente.

```
root@test:~# snort -i eth2 -v
Running in packet dump mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Network Interface eth2
Decoding Ethernet on interface eth2

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.8.5.2 (Build 121)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/snort/snort-
team
```

```
           Copyright (C) 1998-2009 Sourcefire, Inc., et al.
           Using PCRE version: 8.02 2010-03-19

Not Using PCAP_FRAMES
02/04-20:29:18.244984 172.16.0.50 -> 8.8.8.8
ICMP TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:84 DF
Type:8  Code:0  ID:3769   Seq:1  ECHO
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

02/04-20:29:18.316900 8.8.8.8 -> 172.16.0.50
ICMP TTL:47 TOS:0x0 ID:0 IpLen:20 DgmLen:84
Type:0  Code:0  ID:3769  Seq:1  ECHO REPLY
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
       ...
       ...
```

Para visualizar ***BASE*** es necesario abrir el contexto ***acidbase*** en la dirección IP donde estén instalados ***Apache*** y ***BASE.*** Inicialmente aparecerá un mensaje indicando que ***BASE*** aún no ha sido configurado y es necesario conectar con la base de datos de ***Snort*** para crear las tablas necesarias. Las siguientes son algunas de las pantallas que aparecerán al configurar la base de datos. Es muy sencillo y no tiene ninguna pérdida.

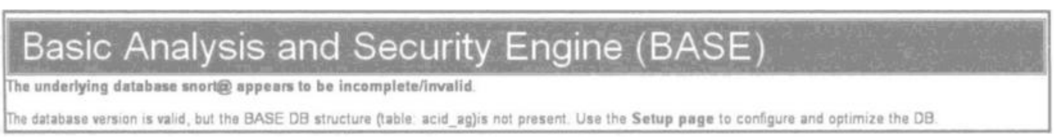


Fig 4.25: Error de configuración de la base de datos.

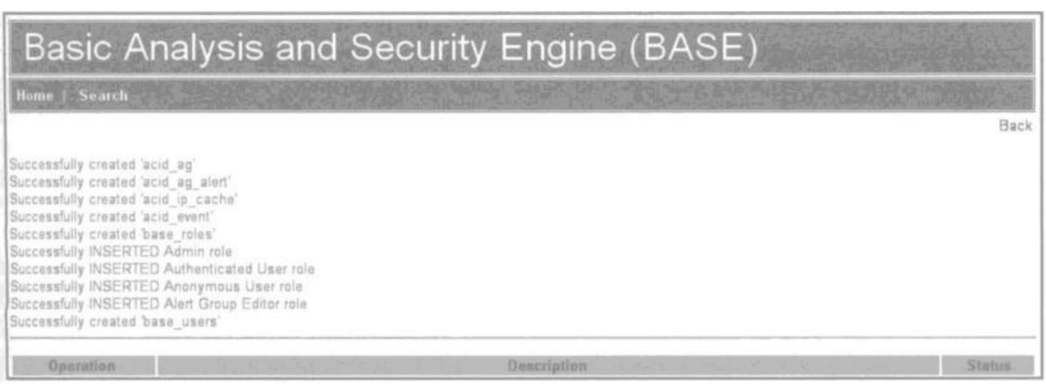


Fig 4.26: Habilitar el modo promiscuo de la interfaz elegida.

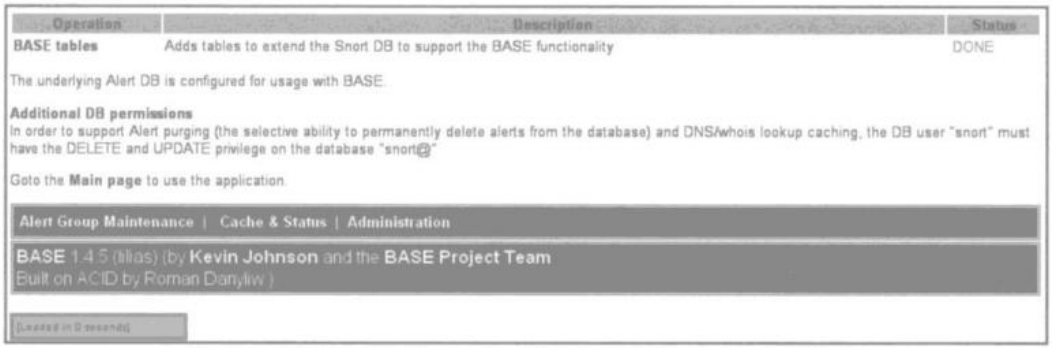


Fig 4.26: Habilitar el modo promiscuo de la interfaz elegida (continuación).

Una vez que todo esté correctamente configurado podrán visualizarse las alertas desde ***BASE*** de diferentes modos. Ya sea por direcciones, protocolos, tipos de detecciones, etc. En la siguiente captura se observan una serie de ataques detectados por ***Snort*** en el entorno de pruebas.

Fig 4.27: Ataques visualizados desde BASE.

Desde una de las máquinas se han realizado escaneos de puertos en busca de servicios, ataques por fuerza bruta a un proxy sip y accesos no autorizados a un recurso web protegido. Además de esos resultados, con la configuración inicial y sin afinar se obtienen muchos falsos positivos. Tan solo hay que afinar las reglas que para que poco a poco el ***IDS*** se vaya adaptando al entorno.

En este instante, además de hacer un poco de ***tuning*** a la configuración, habría que actualizar las reglas de ***Snort*** por unas más actualizadas. Dichas reglas, conocidas como reglas ***VRT***, ***Vulnerability Research Team***, se ofrecen a través de la web de ***Snort*** en tres formatos. Para usuarios suscritos y que por lo tanto están pagando una cuota anual o mensual, usuarios registrados y usuarios anónimos. Una buena opción para empezar con ***Snort*** es la de usuarios registrados, ya que es gratuito y la única limitación que posee la descarga de las reglas es que solo se podrá realizar 30 días después de que se libere el paquete de nuevas reglas. Para poder descargar las reglas, tanto usuarios suscritos como registrados, estos deberán generar el llamado ***Oinkcode*** desde su panel de usuario que permitirá la descarga de las reglas desde aplicaciones automáticas.

Como se comentab, existe un script llamado ***oinkmaster*** incluido en el paquete de ***Snort.*** Con el cual es posible actualizar las reglas que utilizará ***Snort.*** El problema está en que el equipo de ***Snort*** deja de dar soporte para nuevas reglas para versiones de ***Snort*** consideradas como obsoletas. Lógicamente siempre se pueden instalar versiones actualizadas de ***Snort*** a través de los paquetes ofrecidos en la web, pero en el caso de Debian tocará compilar las fuentes para disfrutar de las últimas versiones.

Capítulo V
Protección de la capa de aplicación

Durante los capítulos anteriores se habían diferenciado y acotado tres áreas de un entorno para mejorar su protección. Conseguirlo habrá supuesto utilizar tan solo algunos métodos, la agrupación de varios e incluso todos; todo ello dependiente como de costumbre del tipo de entorno y sus necesidades.

El objeto de este capítulo tiene como finalidad tratar de establecer algunas protecciones adicionales en el sistema operativo ***Linux*** para fortificar determinados aspectos. Sin duda se trata de un capítulo más difuso y variopinto por la temática de los conceptos tratados. La idea principal consiste en configurar de forma adecuada funcionalidades necesarias y comunes de un sistema ***Linux*** para mejorar su seguridad. Los temas a tratar variarán desde permisos de fichero hasta sistemas de detección de intrusos de hosts, pasando por protección de recursos de usuarios entre otros.

1. Jaulas con chroot

Cualquier máquina que ofrezca un servicio accesible desde el exterior se expone por el simple hecho de publicar algún recurso. A pesar de que se realice una buena configuración del servicio, siempre existe alguna posibilidad de que algún atacante pueda sacar provecho de alguna situación para comprometer un sistema. Un ejemplo muy claro de estos casos sucede cuando una aplicación tenga un ***0-day.*** Obviamente, al tratarse de un fallo de seguridad en la aplicación el atacante sabrá cómo actuar en cada caso, utilizando cualquier tipo de ***exploit*** que aproveche la vulnerabilidad.

En caso de llegar a una situación de este tipo existen muchas posibilidades para tratar de disminuir el impacto de los posibles ataques. Hoy en día, la técnica más usada por facilidad de configuración, flexibilidad, coste, etc. es la virtualización de los sistemas operativos. Es decir, se crean máquinas con roles muy concretos para así evitar afectar a otros servicios, ficheros e información sensible en general.

A pesar de ello, aquí se está tratando la creación de capas en un sistema operativo instalado y listo para configurar. En concreto ahora se expone el uso de la creación de jaulas para el aislamiento de aplicaciones, usuarios, servicios, etc. A este tipo de acciones, en los que se aísla algún recurso a modo de estudio o prevención se le conoce como ***sandbox*** o caja de arena. Dicho concepto es

aplicable igualmente al uso de virtualización pues realmente se está aislando la ejecución de un proceso o exposición de sus recursos.

En este caso, el concepto de enjaular un recurso o acceso se llevará a cabo con la aplicación ***chroot*** cuyo nombre proviene de ***Change Root.*** Como su nombre hace pensar, con ***chroot*** se modifica la raíz del sistema para un determinado proceso, haciéndole creer que realmente esa es la raíz de ficheros del sistema. Esto supone que el proceso ejecutado dentro de una jaula ***chroot*** "verá" solo lo que se quiera que vea. Inicialmente una jaula es una carcasa totalmente vacía, sin contenido alguno y será necesario poner en ella los componentes justos y necesarios para permitir la ejecución de una aplicación. Antes de pasar a un ejemplo de ***chroot*** es necesario conocer algunas particularidades y recomendaciones para su uso:

- Nunca utilizar ***chroot*** para un proceso ejecutado como ***root.*** No tiene ningún sentido hacerlo así pues el usuario ***root*** siempre podrá salir de la jaula al sistema de ficheros real.
- Incluir en la jaula los componentes mínimos necesarios para la ejecución de un proceso. En el ejemplo se verá cómo conocer dichos componentes y cómo agregarlos a la jaula.
- Mantener los permisos de los ficheros lo más restrictivos posibles para no conceder a un potencial atacante ni el mínimo ápice de oportunidad para modificar ficheros a su antojo, ejecutables, etc.
- Nunca utilizar una jaula ***chroot*** como único método de seguridad implementado. No servirá de nada en absoluto. Como se comentaba anteriormente se trata de una capa más a establecer para la protección de un entorno.

Esas son algunas de las ideas que se deben tener claras a la hora de implementar una jaula con ***chroot***, a continuación se muestra un ejemplo básico para entender con la práctica el concepto de jaula.

Prueba de concepto de una jaula con chroot

La finalidad de este ejemplo básico consiste en demostrar qué se puede hacer con ***chroot***, poniendo como ejemplo la creación de un entorno enjaulado para la ejecución de un par de aplicaciones básicas.

Preparando el entorno

En primer lugar es necesario crear un directorio de trabajo donde operará ***chroot.*** A este directorio se le suele llamar ***fakeroot.*** Por ejemplo se puede crear el directorio ***/var/chroot.*** Ni que decir tiene, el directorio de trabajo no tiene por qué estar en la partición de la raíz; es posible utilizar otra partición, volúmenes lógicos, etc. Para esta prueba se creará un usuario básico y el directorio base.

```
# useradd carlos
# mkdir /var/newroot
```

El próximo paso a realizar es ejecutar ***chroot*** e indicarle que utilice ese nuevo directorio como base para crear la nueva raíz. Por defecto y en la mayoría de las distribuciones ***chroot*** estará disponible, pues forma parte del paquete ***coreutils.***

Uno de las recomendaciones al utilizar un ***chroot*** consiste en crear la nueva raíz estando posicionados dentro del nuevo directorio, por lo que habrá que ejecutar los siguientes comandos.

```
# cd/var/newroot
# chroot ./
chroot: failed to run command `/bin/bash': No such file or directory
```

Como puede observarse ha aparecido un error. Esto es debido a que cuando ***chroot*** inicia trata de lanzar una ***shell*** para establecer un entorno de trabajo. En estos instantes el directorio base del ***chroot*** está completamente vacío, por lo que hay que crear una serie de directorios básicos.

Primero se empieza por crear el directorio /***bin*** para posteriormente copiar el fichero binario ***bash*** que proporcionará la ***shell.*** Aunque no es todo, para que ***bash*** funcione adecuadamente es necesario copiar las bibliotecas compartidas que utilice. Para este último cometido es necesario averiguar dichas bibliotecas con la utilidad ***Idd.***

```
root@squeetest:~# ldd /bin/bash
        linux-gate.so.1 =>  (0xb7783000)
        libncurses.so.5 => /lib/libncurses.so.5 (0xb7742000)
        libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb773e000)
        libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb75f6000)
        /lib/ld-linux.so.2 (0xb7784000)
```

Por lo tanto, además de copiar el binario ***bash*** será necesario incluir en la estructura ***chroot*** las bibliotecas indicadas por ***Idd.*** Para realizar ambas acciones se puede ejecutar las siguientes instrucciones.

```
# mkdir /var/chroot/bin
# mkdir -p /var/chroot/lib/i686/cmov
# cp /bin/bash /var/chroot/bin
# cp /lib/libncurses.so.5 /var/chroot/lib
# cp /lib/i686/cmov/libdl.so.2 /var/chroot/lib/i686/cmov
# cp /lib/i686/cmov/libc.so.6 /var/chroot/lib/i686/cmov
# cp /lib/ld-linux.so.2 /var/chroot/lib
```

Ahora que todo está preparado llega el momento de probar si ***chroot*** termina mostrando un ***prompt.*** Para ello se utilizará ***chroot*** lanzándolo como el usuario ***carlos*** creado anteriormente. El parámetro

de ***chroot*** que se utiliza en este caso es ***—userspec=<usuario>:<grupo>.*** En este caso el ***uid*** y ***gid*** de ***carlos*** es ***1001.***

```
# chroot --userspec=1001:1001 /var/chroot/
bash-4.1$
```

Como puede observarse, ***chroot*** pone a disposición una ***shell*** básica con ***bash***, por lo que las variables típicas utilizadas en ***bash*** estarán disponibles. Comprobando en este caso el ***PATH*** del usuario se obtendría el siguiente resultado.

```
bash-4.1$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

De acuerdo, ahora llega el momento de comprobar la estructura de ficheros y verificar si realmente el usuario y la ***shell*** ejecutada está en la nueva estructura de directorios enjaulada.

```
bash-4.1$ ls
bash: ls: command not found
```

Como era de esperar no es posible utilizar el comando ***Is*** para realizar el listado de ficheros y directorios pues su ejecutable no está incluido en el entorno enjaulado. El único binario existente en la jaula es el de ***bash***, que se introdujo para hacer funcionar el entorno. Sin embargo, al no estar disponible el comando ***Is***, todo hace indicar que efectivamente la ***shell*** está ejecutándose en la jaula. A partir de este momento quedaría preparar un entorno funcional según las necesidades de la jaula. En este caso de ejemplo se incluirán los ejecutables ***Is*** y ***whoami.*** Como en el caso de ***bash***, será necesario averiguar las dependencias de ambos comandos para incluirlos igualmente en el entorno ***chroot.***

```
# ldd /bin/ls
        linux-gate.so.1 =>  (0xb771e000)
        libselinux.so.1 => /lib/libselinux.so.1 (0xb76fc000)
        librt.so.1 => /lib/i686/cmov/librt.so.1 (0xb76f3000)
        libacl.so.1 => /lib/libacl.so.1 (0xb76eb000)
        libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb75a4000)
        libdl.so.2 => /lib/i686/cmov/libdl.so.2 (0xb75a0000)
        /lib/ld-linux.so.2 (0xb771f000)
        libpthread.so.0 => /lib/i686/cmov/libpthread.so.0 (0xb7587000)
        libattr.so.1 => /lib/libattr.so.1 (0xb7582000)
# ldd /usr/bin/whoami
        linux-gate.so.1 =>  (0xb76f9000)
```

```
        libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb75ab000)
        /lib/ld-linux.so.2 (0xb76fa000)
```

Una vez añadidos los binarios mencionados, las bibliotecas necesarias y comprobando de nuevo el funcionamiento de ambos comandos se obtendría el siguiente resultado.

```
bash-4.1$ pwd
/
bash-4.1$ ls -l
total 16
drwxr-xr-x 2 0 0 4096 Feb  9 18:01 bin
drwxr-xr-x 3 0 0 4096 Feb  9 18:03 lib
drwxr-xr-x 3 0 0 4096 Feb  9 18:02 usr
bash-4.1$ whoami
whoami: cannot find name for user ID 1001
```

La salida del comando ***Is*** demuestra que efectivamente se trata de un entorno aislado con información y funcionalidad limitada. El caso de la información limitada queda reflejado en la ejecución del comando ***whoami***, que ha sido incapaz de resolver el nombre del usuario para el id 1001. Esto es porque en la jaula no se ha incluido el fichero ***/etc/passwd.***

Lo que se ha tratado de demostrar en este punto es el nivel de aislamiento que se puede llegar a conseguir con una jaula utilizando ***chroot.*** En la práctica, realizar una jaula utilizando el método anterior resulta pesado y es por ello que muchas aplicaciones y servicios vienen preparados para crear un entorno ***chroot*** por ellos mismos. Los casos más comunes de servicios utilizados en entornos enjaulados son ***Apache*** y ***Bind*** mediante sus ficheros de configuración y módulos.

2. Permisos especiales, atributos y ACL

La correcta asignación de permisos en determinados ficheros de un sistema puede resultar crucial a la hora de proteger su contenido. Una incorrecta asignación de permisos, grupos, etc. podría revelar información sensible en caso que alguna persona no autorizada pudiera acceder a una máquina como usuario sin privilegios.

Un poco de teoría básica de permisos

Los sistemas basados en ***UNIX*** poseen un sistema de permisos básico habilitado en todos los sistemas de ficheros utilizables. Dichos permisos permiten la asignación a nivel de usuario, grupos u otros;

con los cuales se podrán controlar diferentes accesos a los ficheros como son lectura (r), escritura (w) y ejecución (x). Las combinaciones entre tipos de acceso y niveles de asignación da lugar a los conocidos valores en octal como pueden ser ***640***, ***700***, etc.

Para la gestión de los permisos se utiliza la herramienta ***chmod*** que permitirá modificar los permisos de ficheros y directorios. Para el caso de la visualización de los mismos es posible utilizar el comando ***Is.*** A continuación, algunos ejemplos de permisos de ficheros.

```
# ls -al /etc/passwd
-rw-r--r-- 1 root root 996 feb  9 18:36 /etc/passwd
```

En este caso se ha obtenido la información del importante fichero ***/etc/passwd***', en cuyo caso se puede extraer la siguiente información.

- Tipo de objeto: fichero, indicado por el primer guión o ausencia de letra.
- Permisos del objeto.
  - Permisos para el propietario, primer trío de caracteres: ***rw-,*** lectura y escritura.
  - Permisos para el grupo, segundo trío de caracteres: ***r***—, lectura.
  - Permisos para otros, tercer trío de caracteres: ***r***—, lectura.
- Propietario del fichero: ***root.***
- Grupo asignado al fichero: ***root.***

La notación en octal para el fichero de ejemplo es ***644.*** Se obtiene de ***4+2+0, 4+0+0***, ***4+0+0.*** Para asignar los permisos actuales al fichero ***/etc/passwd*** se podría haber utilizado la siguiente expresión de ***chmod.***

```
# chmod 640 /etc/passwd
```

Permisos especiales

Existen una serie de permisos no mencionados anteriormente, pero que son totalmente estándar y se pueden utilizar sin ninguna modificación u operación adicional. Lo único necesario para ello es la herramienta ***chmod.***

En realidad, a lo comentado anteriormente de la expresión en octal habría que realizarle una corrección. No se expresa mediante una cifra de 3 dígitos, sino de 4. Sin embargo, al tratarse de permisos menos utilizados suele obviarse el primer dígito cuando se mencionan los permisos de un fichero. Unos ejemplos de representación octal de forma totalmente correcta podrían ser ***0750, 4755,*** etc.

Los permisos especiales son tres y en su representación mediante un listado de ficheros con ***Is*** no se agrega ningún campo más para mostrarlos. En su lugar, se visualizan en la posición del permiso de ejecución, por lo que para diferenciar si el fichero era ejecutable o no, las letras de los permisos irán

en minúscula o mayúscula respectivamente. El nombre de los permisos junto con su funcionalidad son los siguientes:

Permiso	Ámbito	Representación (ejecutable / no ejecutable)
Set User ID, setuid	usuario	s / S
Set Group ID, setgid	grupo	s / S
Sticky Bit, STI	otros	t / T

Enumeración de permisos especiales.

Un ejemplo claro de este tipo de permisos en Debian se encuentran en el fichero ejecutable ***/bin/su*** y el directorio ***/tmp.***

```
-rwsr-xr-x 1 root root 29152 feb 15  2011 /bin/su


drwxrwxrwt  4 root root  4096 feb 10 01:17 tmp
```

El fichero ejecutable tiene activado el bit ***setuid***y su representación en octal será ***4755.*** Por su parte, el directorio ***/tmp*** dispondrá de la representación 7777. Ahora bien, ¿qué función tiene cada uno de estos permisos especiales?

- ***setuid.*** Cuando un ejecutable tenga activado este bit hará que el usuario que lo ejecute posea los mismos privilegios que el usuario propietario del ejecutable. Obviamente esos permisos solo los tendrá durante la ejecución de ese fichero. Posteriormente se verá un ejemplo para comprobar el funcionamiento.
- ***setgid.*** Funciona del mismo modo que ***setuid*** pero esta vez atendiendo al grupo de pertenencia de un ejecutable.
- ***STI.*** Es un bit especial que suele colocarse en directorios con permisos de acceso total, es decir lectura, escritura y ejecución para propietario, grupo propietario y otros. Su funcionalidad hace que solo se puedan eliminar y modificar ficheros por los propietarios de los mismos y no por otro usuario, independientemente de los permisos ya comentados del directorio. Como se mostraba, un ejemplo para este tipo de permiso se encuentra en el directorio ***/tmp***\ donde son muchas las aplicaciones que pueden escribir y esto proporcionará protección evitando eliminaciones o sobre-escrituras indebidas.

Un ejemplo de ***setuid*** claro se encuentra en el comando ***passwd***, cuyos permisos son ***4755.*** Con ***passwd*** cualquier usuario puede modificar su contraseña de ***login***, por lo que en una configuración de autenticación por defecto siempre se modificará el fichero ***/etc/shadow.*** Sin embargo ***/etc/shadow*** dispone de permisos ***0640*** con propietario y grupo ***root***, por lo que solo es accesible y modificable por el usuario ***root.*** Es por ello que ***passwd*** necesita del bit ***setuid.*** Si se elimina el ***setuid*** para ***passwd***, a los usuarios no privilegiados les será imposible modificar su ***password.***

Otro ejemplo muy visual de ***setuid*** puede comprobarse estableciéndolo en el comando ***id.***

```
# ls -al /usr/bin/id
-rwxr-xr-x 1 root root 26708 abr 28  2010 /usr/bin/id
# su carlos -
$ id
uid=1001(carlos) gid=1001(carlos) grupos=1001(carlos)
$ exit
# chmod u+s /usr/bin/id
# ls -al /usr/bin/id
-rwsr-xr-x 1 root root 26708 abr 28  2010 /usr/bin/id
# su carlos -
$ id
uid=1001(carlos) gid=1001(carlos) euid=0(root) grupos=0(root),1001(carlos)
```

En la secuencia de comandos anterior se ha establecido el ***setuid*** para el comando ***id*** y se ha comprobado el efecto para la ejecución con un usuario no privilegiado. La salida de ***id*** habla por sí sola, pudiendo comprobar como el ***euid*** o ***uid*** efectivo posee el valor de ***root.***

Como nota adicional cabe destacar que ***setuid*** y ***setgid*** no surten efecto cuando se aplican a Scripts interpretados como puede ser ***bash*** por ejemplo. Es un comportamiento establecido por defecto para evitar problemas de seguridad causados por ***scripts*** mal programados.

Atributos

Además de los permisos ya mencionados, existen una serie de características o atributos disponibles para los ficheros. Dichos atributos permitirán que los ficheros tengan una serie de peculiaridades o comportamientos cuando sean aplicados de manera conveniente.

Para manipular los atributos de los ficheros se utiliza la aplicación ***chattr***, incluida por defecto en la mayoría de distribuciones. En el caso de Debían está disponible con el paquete ***e2fsprogs.*** Si se desean visualizar los atributos es posible utilizar ***Isattr.*** Por defecto al crear un fichero, este no dispone de ningún atributo asignado salvo en sistemas ***ext4***, en los que se añaden a ciertos ficheros el atributo ***e***, ***extend*** que mejora el tiempo de acceso a los mismos.

```
# lsattr /etc/passwd
---------------------- /etc/passwd
```

A continuación se explican dos de estos parámetros que pueden resultar interesantes, atendiendo sobre todo a la seguridad sobre ficheros:

- /, ***inmutable*** es un parámetro utilizado para evitar la sobre-escritura o eliminación de un fichero. Ni siquiera el usuario ***root*** dispone de privilegios para ello. Si el fichero dispone de este atributo, la funcionalidad comentada estará habilitada.
- ***a***, ***append*** se utiliza para indicar que al fichero solo se le puede agregar contenido pero no sobre-escribir dicho contenido. Sin duda una opción muy interesante para proteger ***logs*** importantes del sistema como ***auth.log.***

A modo de prueba de ambos atributos se creará un fichero y se le asignará el atributo ***i.*** Acto seguido se intentará eliminar el fichero.

```
# chattr +i prueba.txt
# lsattr prueba.txt
----i-----------e- prueba.txt
# rm prueba.txt
rm: no se puede borrar «prueba.txt»: Operación no permitida
# echo "Otra línea" >> prueba.txt
-bash: prueba.txt: Permiso denegado
```

Como se comentaba, ni siquiera el usuario ***root*** puede eliminar el fichero y tampoco añadir contenido cuando el atributo ***i*** está activo.

Para comprobar el atributo ***a***, el fichero ***prueba.txt*** ya dispone de contenido y sus atributos han sido totalmente limpiados. Una vez establecido el atributo ***append*** el comportamiento será como el siguiente.

```
# cat prueba.txt
Primera línea de prueba.
# chattr +a prueba.txt
# lsattr prueba.txt
-----a-----------e- prueba.txt
# echo "Machacando el fichero" > prueba.txt
-bash: prueba.txt: Operación no permitida
# echo "Añadiendo otra línea" >> prueba.txt
# cat prueba.txt
Primera línea de prueba.
Añadiendo otra línea
```

Se verifica que se puede agregar contenido al fichero pero no modificar el anterior o macharlo.

Sin duda resultan características interesantes y como sucede siempre habrá diversas situaciones donde se le pueda sacar provecho a esta funcionalidad. Por otra parte, existen además otros atributos extendidos que son utilizados para ***SELinux.***

ACL, Access Control List

Uno de los aspectos que quizás se echan de menos en un sistema ***Linux*** es que para toda su jerarquía de ficheros se pueda usar algo más que un usuario y un grupo de usuarios para conceder accesos. Por este motivo se implementaron las ***ACL*** para los diferentes sistemas de ficheros soportados.

Hoy en día la mayoría de las distribuciones dispondrán de soporte para utilizar ***ACL,*** pero si se desea comprobar que efectivamente se pueden implementar ***ACL*** en ficheros es posible utilizar la siguiente orden.

```
# cat /boot/config-2.6.32-5-686 | grep -i _acl
CONFIG_EXT2_FS_POSIX_ACL=y
CONFIG_EXT3_FS_POSIX_ACL=y
CONFIG_EXT4_FS_POSIX_ACL=y
CONFIG_REISERFS_FS_POSIX_ACL=y
...
```

Como se puede observar entre otros, para los sistemas de ficheros ***extX*** está habilitado el soporte paraACL. En caso de que se hubiera obtenido ***n*** como respuesta tocaría recompilar el núcleo para habilitar el soporte.

Antes de comenzar a trabajar con las ***ACL*** es necesario instalar un paquete en el sistema, así como habilitar una opción a la hora de montar la partición o particiones donde se desee utilizar ***ACL***. El paquete a instalar es ***acl***, que contendrá las utilidades necesarias para manipular las ***ACL*** de los ficheros. Por otra parte, en este caso habrá que habilitar el soporte en la partición raíz modificando la línea en el fichero ***/etc/fstab*** del siguiente modo.

```
/dev/sda1      /       ext4    errors=remount-ro,acl 0       1
```

Es decir, se añade la palabra clave ***acl*** al grupo de opciones. Para hacer que la partición disponga de soporte para ***acl*** en este instante sin tener que reiniciar la máquina podría ejecutarse la siguiente orden.

```
# mount -o remount,acl /dev/sda1
# mount
/dev/sda1 on / type ext4 (rw,errors=remount-ro,acl)
```

Se puede verificar con el comando *mount* que efectivamente la partición raíz está montada con soporte para *acl*, por lo que es hora de empezar a probar su funcionalidad con los siguientes comandos.

- *setacl*. Este comando se utiliza para establecer las *ACL* en los ficheros, ya sean estos directorios o ficheros como tal.
- *getacl*. Usado para obtener información de las *ACL* aplicadas a un fichero o conjunto de ellos.
- *chacl*. Permite realizar modificaciones en las *ACL* aplicadas a ficheros y directorios.

Otro aspecto a tener en cuenta para las entradas de una *ACL* existen varios tipos, dependiendo de a qué se le apliquen y cómo se apliquen.

- usuario. Permite definir el acceso para un usuario
- grupo. Especifica el acceso para un grupo.
- otro. Define el acceso para otros usuarios.
- máscara. Se pueden utilizar entradas *ACL* de tipo máscara para establecer los permisos efectivos que tendrán todas las entradas *ACL*. Nunca añade privilegios a las entradas *ACL* de otros tipos, sino que restringe.
- default. Se tratan de entradas *ACL* únicamente aplicables a directorios. Dichas entradas se aplicarán por defecto a los ficheros contenidos dentro del directorio.

Por otra parte, como se mostrará en el ejemplo, las entradas de una *ACL* se especifican mediante tres campos separados por el símbolo de dos puntos. Generalizando la estructura, una entrada *ACL* tendría el siguiente aspecto.

```
<tipo>:<usuario/grupo>:<permisos>
```

La especificación de usuario o grupo se hará solo cuando el tipo sea usuario o grupo. Con respecto a los permisos, se especificarán del modo *rwx*, sustituyendo por un guion en caso de ausencia de uno de los permisos.

Ejemplo de ACL

Para que se vea más claro se realizarán una serie de ejemplos para verificar cómo afecta el establecimiento de la *ACL* a los permisos sobre un fichero.

Se supone que el usuario *root* crea un fichero en */tmp* con unos permisos básicos.

```
# echo "Prueba de ACL" > /tmp/test
# chmod 640 /tmp/test
# ls -al /tmp/test
-rw-r----- 1 root root 31 feb 10 18:24 /tmp/test
```

Si por ejemplo dos usuarios no propietarios del fichero tratasen de realizar operaciones sobre él obtendrían mensajes denegando el acceso.

```
carlos@squeetest:/root$ cat /tmp/test
cat: /tmp/test: Permiso denegado
carlos@squeetest:/root$ echo "Tratando de escribir" >> /tmp/test
bash: /tmp/test: Permiso denegado
```

Este comportamiento puede modificarse estableciendo una serie de entradas en la ACL. Se crearán dos, una para el usuario ***cavíos*** y otra para el usuario ***pablo.*** Cada una de ella proporcionará diferentes accesos al fichero. Por su parte el usuario ***carlos*** obtendrá permisos de lectura y escritura mientras que el usuario ***pablo*** solo podrá leer el fichero.

```
squeetest:~# setfacl -m u:carlos:rw- /tmp/test
squeetest:~# setfacl -m u:pablo:r-- /tmp/test
squeetest:~# getfacl /tmp/test
getfacl: Eliminando '/' inicial en nombres de ruta absolutos
# file: tmp/test
# owner: root
# group: root
user::rw-
user:carlos:rw-
user:pablo:r--
group::r--
mask::rw-
other::---
```

Una vez aplicadas las reglas se procede a realizar un listado de la ***ACL*** asociada al fichero ***/tmp/test*** con la aplicación ***getfacl.*** El resultado muestra que existen dos entradas, una para el usuario ***carlos*** y otra para el usuario ***pablo.*** Por otra parte se ha establecido la máscara de forma automática con los permisos más permisivos de todas las entradas, en este caso el ***rw-*** de ***carlos.***

Las tres últimas entradas, cuyo segundo campo está vacío, reciben el nombre de entradas ***ACL*** base y se corresponden con los permisos básicos asociados al fichero.
Ahora que la ***ACL*** está establecida se comprueba si efectivamente ambos usuarios disponen de los permisos especificados.

```
carlos@squeetest:~$ echo "Línea de Carlos" >> /tmp/test
carlos@squeetest:~$ cat /tmp/test
```

```
Prueba de ACL
Línea de Carlos
pablo@squeetest:~$ echo "Línea de Pablo" >> /tmp/test
bash: /tmp/test: Permiso denegado
pablo@squeetest:~$ cat /tmp/test
Prueba de ACL
Línea de Carlos
```

Es posible saber si a un fichero o directorio se le está aplicando una ***ACL.*** Cuando se realiza un listado con ***Is***, un fichero o directorio con ***ACL*** se listará con un símbolo + al final de la cadena de permisos asociados. Por otra parte, los tres caracteres intermedios, correspondientes a los permisos de grupo, tomarán el valor de la máscara establecida en la ***ACL.***

```
# ls -al /tmp/test
-rw-rw----+ 1 root root 31 feb 10 18:46 /tmp/test
```

Es posible modificar la máscara del siguiente modo para comprobar cómo afectaría al usuario ***Carlos*** en este caso.

```
squeetest:~# setfacl -m m::r-- /tmp/test
squeetest:~# getfacl /tmp/test
getfacl: Eliminando '/' inicial en nombres de ruta absolutos
# file: tmp/test
# owner: root
# group: root
user::rw-
user:carlos:rw-                 #effective:r--
user:pablo:r--
group::r--
mask::r--
other::---

squeetest:~# ls -al /tmp/test
-rw-r-----+ 1 root root 31 feb 10 18:46 /tmp/test
```

Si el usuario ***cavíos*** tratara de modificar el fichero ***/tmp/test*** en estos instantes obtendría un error, pues la máscara está impidiendo su acceso al fichero como escritura.

```
carlos@squeetest:~$ echo "Otra línea de Carlos" >> /tmp/test
bash: /tmp/test: Permiso denegado
```

Llegados a este punto, puede concluirse que no solo existen los típicos permisos de ficheros para usuarios, grupos y otros establecidos con ***chmod.*** Como se ha podido comprobar existen más opciones que permitirán establecer una capa adicional en el sistema de ficheros para accesos no autorizados o simplemente disponer de un control más exhaustivo del propio sistema de ficheros.

3. Elevación de privilegios con 'sudo'

En numerosas ocasiones, en entornos gestionados por muchos operadores resulta muy complicado e inseguro tener que ofrecer acceso total a una máquina. Por supuesto ello implica dar a conocer el ***password*** de ***root*** a ese grupo de técnicos u operadores encargados de ciertas labores de mantenimiento de un sistema. Sin embargo existe una opción cada vez más utilizada para que la mencionada delegación, reparto, etc. de tareas no deba suponer una brecha de seguridad por el hecho de que muchas personas conozcan las credenciales de ***root.***

La opción comentada consiste en utilizar ***sudo***, que recibe el nombre por ***Super User Do.*** Se trata de una herramienta con la que un determinado usuario puede realizar tareas en nombre de otro y por ende con los permisos de este último. Normalmente el usuario simulado, por decirlo de algún modo, es ***root.*** Para limitar las funcionalidades de un usuario que adquiera temporalmente privilegios de otro usuario o de ***root***, se establecen una serie de políticas o directivas donde se controlará qué puede o qué no puede hacer cada usuario. Es decir, la finalidad principal consiste en conceder privilegios administrativos sobre una máquina de forma controlada.

Instalación de sudo y análisis de sus componentes

A día de hoy todas las distribuciones incluyen en sus repositorios la aplicación ***sudo***, e incluso muchas de ellas la instalan por defecto. En el caso de Debían, dependiendo de cómo se haya instalado el sistema estará disponible desde el inicio. Es decir, a la hora de instalar el sistema se solicita la elección de la utilización del usuario ***root*** de forma tradicional o la administración de la máquina a través de ***sudo.*** En caso de no estar instalada, para Debían puede obtenerse del siguiente modo.

```
# apt-get install sudo
```

Al finalizar la instalación se habrán instalado los componentes necesarios para utilizar ***sudo.*** Entre todos los componentes, los más interesantes son los siguientes:

- ***sudo.*** Es la aplicación principal que se utilizará para realizar acciones en nombre de otros usuarios del sistema. Como se comentaba anteriormente, por norma general suele utilizarse para que el usuario que invoque sudo ejecute la acción especificada como usuario ***root.***

- ***/etc/sudoers.*** Es el fichero donde se declaran las opciones o límites para el uso de ***sudo.*** Como se verá posteriormente el fichero dispone de varias secciones bien diferenciadas donde se declararán diferentes tipos de directivas o variables. Los permisos de este fichero son ***0440*** y su propietario es ***root.***

- ***visudo.*** El comando con el que se debería editar el fichero de configuración ***/etc/sudoers,*** pues evita que otro usuario pueda editar el fichero paralelamente provocando conflictos. Por otra parte, al salvar la configuración realizará comprobaciones en busca de errores sintácticos. Se utilizará el editor establecido por defecto en el sistema y en el caso de Debian puede comprobarse con el comando ***update-alternatives —list editor.*** En una instalación por defecto de Debian el editor por defecto será ***nano.***

sudo

El comando sudo suele utilizarse del modo ***sudo <acción a ejecutar>.*** Al no indicar ningún tipo de parámetro, esa ***<acción a ejecutar>*** se lanzará con privilegios de usuario ***root.*** Todo ello por supuesto siempre que el usuario que está ejecutando ***sudo*** esté habilitado en el fichero de configuración ***/etc/sudoers***, de modo que pueda utilizar ***sudo*** haciéndose pasar por el usuario ***root.*** Cuando se ejecuta ***sudo*** se solicita el***password*** del usuario que lo ejecuta. Dicha introducción de credenciales, si resulta exitosa, creará una sesión durante 15 minutos que permitirá ejecutar a ese usuario las siguientes acciones con ***sudo*** sin introducir nuevamente el ***password.***

Existen una serie de parámetros que modifican el comportamiento por defecto de ***sudo.*** De todos ellos, algunos que pueden resultar interesantes son:

- ***-u,*** permite especificar el usuario en cuyo nombre se vaya a ejecutar una acción. Sin este parámetro se lanzaría la acción en nombre de ***root.*** Puede indicarse tanto el nombre del usuario como el ***uid*** precedido de una almohadilla, ***#<uid>.***

- ***-g,*** tiene el mismo cometido que ***-u*** pero orientado a los grupos de usuario. De igual modo puede especificarse el ***gid*** utilizando ***#<gid>.***

- ***-k,*** es un parámetro que se utiliza para invalidar una sesión ***sudo.*** Es decir, tras ejecutar ***sudo -k,*** cuando el usuario desee ejecutar nuevamente ***sudo*** con una acción se le solicitará su ***password.***

- -/, su uso sin ningún otro parámetro muestra por pantalla las acciones o comandos permitidos o prohibidos por el usuario que ejecuta ***sudo.*** Puede combinarse con ***-U*** para ver el listado de otro usuario.

- -s / ***-i,*** permiten invocar una ***shell*** en caso de que no indicar más parámetros.

El siguiente fragmento de código muestra una serie de ejemplos con algunos de los parámetros. Se utiliza al usuario ***Carlos,*** estando este autorizado en el fichero ***/etc/sudoers*** para ejecutar acciones como ***root.***

```
carlos@squeetest:~$ sudo -l
[sudo] password for carlos:
```

```
Matching Defaults entries for carlos on this host:
    env_reset
User carlos may run the following commands on this host:
    (ALL) ALL

carlos@squeetest:~$ sudo -s
squeetest:/home/carlos# exit
carlos@squeetest:~$ sudo -k
carlos@squeetest:~$ sudo -s
[sudo] password for carlos:
carlos@squeetest:~$ sudo id
[sudo] password for carlos:
uid=0(root) gid=0(root) grupos=0(root)
```

Para el parámetro -/ se han mostrado las características de las que dispone el usuario ***carlos*** utilizando ***sudo*** para ejecutar acciones como ***root.*** Aunque ahora no se entienda muy bien la salida, con ***(ALL) ALL*** se resume que puede ejecutar cualquier aplicación como ***root.*** Posteriormente se ha utilizado -5 para iniciar una ***shell*** como usuario ***root.*** Con ***-k*** se ha eliminado la [6]***cookie***' de ***sudo***, por lo que al ejecutarlo nuevamente junto al comando ***id*** se ha solicitado el ***password*** al usuario ***carlos.*** También resulta interesante la salida del comando ***id*** utilizando ***sudo***, ***\carlos*** es considerado ***rootl***

Fichero /etc/sudoers

Como se comentaba anteriormente, es el fichero en el que se determinan qué accesos se conceden a qué usuarios y con qué opciones. En el caso de Debían es posible fragmentar la configuración de los ficheros como suele ser habitual para muchas otras aplicaciones o servicios. En concreto, todos los ficheros que se establezcan en el directorio ***/etc/sudoers. d*** serán interpretados como configuración de ***sudo*** y agregados de forma lógica al final del fichero ***/etc/sudoers.*** Es importante recordar que para la edición del fichero se recomienda la utilización de la aplicación ***visudo.***

El fichero ***/etc/sudoers*** está dividido por defecto en tres secciones para la facilidad y claridad a la hora de realizar una configuración. A pesar de ello, es posible escribir el fichero sin importar el orden de las opciones. Las secciones en las que se encuentra dividido por defecto, y al mismo tiempo los diferentes tipos de definiciones posibles son:

- ***Defaults*** u opciones. Se tratan de opciones utilizadas para modificar el comportamiento de ***sudo.*** Dichas opciones son aplicables tanto a nivel global, para que afecten a todos los usuarios, grupos o máquinas configurados en ***sudo***, a nivel de usuario, o a nivel de máquina. Estas opciones pueden ser booleanos en cuyo caso se negarán con el símbolo de exclamación (!), de valor o de cadena. Posteriormente mediante un ejemplo se entenderán mejor las opciones y los valores que pueden tener. Tiene el formato ***Defaults <opciónl>,...,<opciónN>*** para

las opciones globales y ***Defaults:<usuario> <opción1<opciónN>*** para las opciones de usuario. Un ejemplo de opciones para un usuario en concreto sería: ***Defaults:carlos insults,*** que mostraría un insulto por la pantalla si el usuario ***carlos*** se equivocase de ***password.***

- ***Alias.*** Se trata del típico concepto de alias. Gracias a esta funcionalidad se podrán crear agrupaciones de elementos referenciados por un nombre o etiqueta. Se pueden crear diferentes tipos de ***alias,*** siendo estos para máquinas ***{Host Alias),*** usuarios ***{User Alias)*** y comandos ***{CmndAlias).*** Para el primer caso por ejemplo se podrá crear el alias de técnicos, en los que se incluirán a determinados usuarios con el rol de técnico. De igual modo ocurrirá con los comandos, donde se podrán agrupar por ejemplo las utilidades típicas para reiniciar o apagar un máquina, etc. La forma de definir los alias ha de seguir el formato ***<Tipo> <NombreAlias> = <Elementol>,...,<ElementoN>[: <NombreAliasN> — <Elemento 1 <ElementoN>].*** Un ejemplo de ***alias*** de comandos sería: ***Cmnd Alias BARRABIN = /bin/.***
- ***ACL.*** Son las reglas definidas. Realmente es donde se autoriza a un usuario, grupo, máquina, o una combinación de ellos a utilizar determinados comandos. La forma en que se define es ***<usuario/%grupo> <host> = [(<runAsUserList>:<runAsGroupList>)] <comando 1 >,...,<comandoN>.*** Cada uno de los campos pueden ser ***alias,*** que es para lo que se definen realmente. Ayudarán a facilitar la creación y mantenimiento de las ***ACL.*** Un ejemplo de ***ACL*** podría ser ***carlos ALL = /sbin,*** que otorgaría al usuario ***carlos*** la posibilidad de utilizar aplicaciones contenidas en ***/sbin.***

El fichero instalado por defecto con ***sudo*** en Debian es el siguiente.

```
Defaults        env_reset

# Alias specification

# User privilege specification
root    ALL=(ALL) ALL

# Allow members of group sudo to execute any command
# (Note that later entries override this, so you might need to move
# it further down)
%sudo ALL=(ALL) ALL
#includedir /etc/sudoers.d
```

Se añade la opción global y booleana ***envjreset,*** cuyo cometido al utilizar ***sudo*** será eliminar todas las variables de entorno del usuario salvo ***LOGNAME, MAIL, SHELL, USER, USERNAME*** y las ***SUDO*.***

Por otra parte existen dos ***ACL,*** una para el usuario ***root*** y otra para el grupo ***sudo.*** En ambos casos se proporciona acceso de ejecución en todas las máquinas, pudiendo ejecutar las órdenes como cualquier usuario y cualquier comando. Para estos casos se utiliza la palabra reservada ***ALL,*** que hace coincidir cualquier valor. Con respecto a la línea del grupo ***sudo,*** es necesario establecer el prefijo ***%*** para indicar que se trata de un grupo. Antes se comentó que al usuario ***carlos*** se le habían asignado todos los permisos para ejecutar cualquier comando como ***root.*** Se realizó agregando al usuario en el grupo ***sudo.***

```
$ id carlos
uid=1001(carlos) gid=1001(carlos) grupos=1001(carlos),27(sudo)
```

Como último comentario acerca del fichero por defecto, cabe mencionar que la directiva ***includedir*** para granular el fichero de configuración se encuentra comentada por defecto. Por otra parte, como se indica en el comentario del fichero, las directivas pueden sobre-escribirse y se respetará el orden en el que se establecen.

Ejemplo de configuración para sudo

Con motivo de entender mejor el funcionamiento de ***sudo*** y de su fichero de configuración se va a establecer una configuración con las siguientes características y configuración del sistema.

1. En la máquina existen 6 usuarios llamados: ***carlos, pablo, manu, oca, german*** y ***juanmi.***
2. Se crearán los grupos de usuarios ***CONTACTOS, TECNICOS y ADMINISTRADORES*** estableciendo usuarios en cada uno de ellos.
3. Se crearán grupos de comandos ***BIN, SBIN*** y ***GESTIONES***; estableciendo comandos o rutas relacionadas.
4. Para los usuarios ***ADMINISTRADORES*** no se solicitará ***password.***
5. Todos los usuarios solo tendrán 1 intento para logarse, con un ***timeout*** de 1 minuto y con un tiempo de sesión sin solicitar nuevamente el ***password*** de 5 minutos. Se continuará usando la opción ***env reset.***
6. El usuario ***german*** será insultado si introduce su ***password*** de forma incorrecta.
7. Los ***CONTACTOS*** podrán ejecutar los comandos bajo el ***alias GESTIONES*** como usuario ***root.***
8. Los ***TECNICOS*** podrán ejecutar todos los comandos bajo el ***alias SBIN,*** agregando adicionalmente el comando ***/bin/date*** y denegando el uso del comando ***/sbin/ifconfig*** y ***/sbin/ip.*** Todo ello como usuario ***root.***
9. Los ***ADMINISTRADORES*** podrán ejecutar todos los comandos bajo los ***alias BIN y SBIN en*** nombre de cualquier usuario.

Para todo ello el fichero de configuración quedaría del siguiente modo.

```
# /etc/sudoers
# OPCIONES -----------------------------------
Defaults        env_reset,timestamp_timeout=5,passwd_timeout=1,\
                passwd_tries=1
Defaults:german insults

# ALIAS --------------------------------------
User_Alias ADMINISTRADORES      = carlos,pablo
User_Alias TECNICOS             = oca,manu
User_Alias CONTACTOS            = german,juanmi

Cmnd_Alias SBIN         = /sbin/,/usr/sbin/,/usr/local/sbin
Cmnd_Alias BIN          = /bin/,/usr/bin/,/usr/local/bin
Cmnd_Alias GESTIONES    = /sbin/halt,/sbin/shutdown,/bin/date

# ACL----------------------------------------
# Root y sudo sigue manteniendo sus privilegios totales
root    ALL=(ALL) ALL
%sudo   ALL=(ALL) ALL

ADMINISTRADORES ALL = (ALL) NOPASSWD: BIN,SBIN
TECNICOS        ALL = SBIN,/bin/date,!/sbin/ifconfig,!/sbin/ip
CONTACTOS       ALL = GESTIONES
```

Se han utilizado tanto valores booleanos como enteros en la configuración anterior. El booleano, correspondiente a un directorio, se ha negado con el símbolo de admiración. Por otro lado para la indicación de todos los binarios de un directorio se especifica la ruta con una / al final. Para evitar la solicitud de ***password*** en el caso de los usuarios ***ADMINISTRADORES*** se utiliza el ***tag NOPASSWD.*** Todo ello y muchos más parámetros están documentados de forma muy detallada en ***man sudo.***

Algunos de los resultados obtenidos tras la configuración anterior son los siguientes.

```
oca@squeetest:~$ sudo ifconfig
[sudo] password for oca:
Sorry, user oca is not allowed to execute '/sbin/ifconfig' as root on squeetest.
<----->
juanmi@squeetest:~$ sudo date
```

```
[sudo] password for juanmi:
Sorry, try again.
sudo: 1 incorrect password attempt
juanmi@squeetest:~$
<----->
german@squeetest:~$ sudo id
[sudo] password for german:
Listen, burrito brains, I don't have time to listen to this trash.
sudo: 1 incorrect password attempt
german@squeetest:~$
```

Otros parámetros interesantes pueden ser el envío de e-mails cuando un usuario no tenga permitida la ejecución de un comando, se equivoca de ***password***, etc.

Ahora que la base está explicada y se dispone del ejemplo anterior, será más sencillo generar una configuración ideal para cada entorno. Como se puede ver, ***sudo*** dispone de mucha flexibilidad y potencia, además de seguridad si se configura de forma correcta y controlando los permisos.

4. Limitación de recursos

En cualquier tipo de entorno, ya sea en producción o de pruebas, resulta muy importante controlar los recursos que se utilizan en una máquina. En numerosas ocasiones las configuraciones por defecto o los límites que se establecen no son suficientes. Es uno de los problemas que suceden más a menudo, cuando servicios o usuarios ponen a prueba los sistemas saturándolos de contenido, utilización de CPU, memoria, etc.

En este punto se expondrán algunos conceptos para tratar de contener a los usuarios y la utilización de recursos. Se verán cuotas y límites de usuario, políticas de ***passwords*** y cómo monitorizar aplicaciones, además de alertar y detenerlas cuando presenten un funcionamiento anómalo y comprometedor para el entorno.

Inicio de sesión, passwords y límites

Cuando un usuario realiza un inicio de sesión en un sistema, el proceso de autenticación puede realizarse de diversos modos; siendo posible la autenticación básica UNIX, mediante un servidor ***LDAP***, ***Kerberos***, etc. Dichos inicios de sesión implican que los usuarios utilicen los mencionados mecanismos de autenticación, que la información proporcionada por el usuario sea la correcta y que

el sistema la considere como válida. Posteriormente, una vez que un usuario ha iniciado sesión, esta se considera como activa siendo registra en el sistema tanto en su inicio como en su fin.

En los sistemas ***Linux*** modernos la autenticación y el control de límites para usuario, en su gran mayoría de aspectos, se realiza mediante ***PAM***, ***Pluggable Authentication Modules.*** Se trata de una capa intermedia entre los diferentes mecanismos de autenticación y las aplicaciones. Su cometido consiste en ofrecer diferentes configuraciones qü^ definirán cómo se autenticarán los usuarios o aplicaciones en un sistema. La principal premisa de ***PAM*** es poder utilizar diferentes métodos de autenticación sin tener que modificar aplicaciones o servicios para ello, tan solo mediante la propia configuración de ***PAM.*** Con respecto a la configuración de ***PAM***, los diferentes módulos que pueden utilizarse se dividen en cuatro categorías según su función:

- ***auth.*** Proporciona la autenticación en sí. Es el paso en el que se determina si las credenciales son correctas. Se puede utilizar desde autenticación por ***password*** almacenado en el sistema, ***Kerberos***, ***LDAP***, tarjetas inteligentes e incluso acercando un dispositivo ***bluetooth*** en combinación con un ***pin.***
- ***account.*** Se verifica si el usuario tiene permitido el acceso en ese instante. Es posible que la cuenta esté bloqueada, que sea un horario no permitido para la entrada en el sistema, etc.
- ***password.*** Los módulos baj o esta categoría se utilizarán para la modificación de ***passwords*** y su control.
- ***session.*** Se realizan acciones varias para que el usuario pueda realizárfunciones una vez que se ha autenticado. Por ejemplo establecer su directorio personal.

Cada uno de los módulos disponibles dentro de cada una de las categorías se utilizan para configurar el comportamiento de ***PAM*** y por lo tanto el proceso de autenticación. Los ficheros principales donde se realiza dicha configuración son ***/etc/pam.conf*** o bien los ficheros contenidos en ***/etc/pam.d/.*** En el caso de Debian, se utiliza la opción de ficheros fragmentados para la configuración de ***PAM***, el nombre de cada fichero identificará el servicio o aplicación que se está configurando. Las entradas de configuración de distintos módulos de ***PAM*** de los ficheros del directorio ***/etc/pam.d/*** tienen la siguiente forma:

```
tipo   control        ruta   [argumentos]
```

- El ***tipo*** puede ser cualquiera de los que se detallaron anteriormente. En principio, todas las entradas de configuración de ***PAM*** del mismo tipo se ejecutan de forma secuencial. Aunque no siempre resulta así, dependiendo del siguiente parámetro.
- El ***control*** indica qué realizar en caso de éxito o fallo. Pueden existir los siguientes valores o expresiones:
  - ***required***, indica que para que la pila de módulos del mismo tipo tenga éxito, este módulo debe tener éxito igualmente. En caso de fallo, este no se notificará hasta que finalice la ejecución de la pila del mismo tipo.
  - ***requisite***, funciona del mismo modo que ***required***, salvo que en caso de error no continúa ejecutándose la pila.

- ***sufficient***, indica que si el módulo tiene éxito no se continuará ejecutando la pila de módulos del mismo tipo. En caso negativo se seguirá con la ejecución de los demás módulos.
- ***optional***, será tenido en cuenta solo si no se ha llegado al estado de éxito o fallo ejecutando la pila.
- [nueva sintaxis]. Se trata de un nuevo método para definir parejas de ***valor=acción.*** Se definen las parejas entre corchetes. Para su explicación se utilizará el ejemplo de la configuración para ***login.***

• La ***ruta*** permite indicar la ubicación del módulo. Como ruta relativa se utiliza el directorio /***lib/s&curity***, sin embargo es posible indicar una ruta absoluta.

• Los ***argumentos*** son propios de cada módulo de ***PAM*** y están documentados en cada uno de sus manuales. Determinarán el funcionamiento de cada módulo modificando su comportamiento.

Por otra parte, en un sistema con autenticación por defecto se utilizan los ficheros ***/etc/passwd*** y /***etc/shadow*** para las cuentas de usuario. Adicionalmente es posible establecer ciertos límites que se verán a continuación con el fichero ***/etc/security/limits.conf*** entre otros.

Configuración por defecto en Debían

A continuación se exponen las configuraciones por defecto de algunos aspectos relacionados con el ***login, passwords*** y cuentas en general.

PAM

Como se comentaba con anterioridad, se expone la configuración por defecto de ***PAM para*** el servicio ***login.*** El fichero ***/etc/pam.d/login*** tiene un aspecto como el siguiente.

```
auth       optional   pam_faildelay.so  delay=3000000
auth [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die]
pam_securetty.so
auth       requisite  pam_nologin.so
session [success=ok ignore=ignore module_unknown=ignore default=bad]

pam_selinux.so close
session       required   pam_env.so readenv=1
session       required   pam_env.so readenv=1 envfile=/etc/default/locale

@include common-auth

auth       optional   pam_group.so
session    required   pam_limits.so
session    optional   pam_lastlog.so
```

```
session    optional   pam_motd.so
session    optional   pam_mail.so standard

@include common-account
@include common-session
@include common-password

session [success=ok ignore=ignore module_unknown=ignore default=bad]

pam_selinux.so open
```

Se puede observar cómo en el fichero aparecen entradas que comienzan por ***@include.*** Es decir, se está incluyendo el contenido de los ficheros ***/etc/pam.d/common-auth***, ***/etc/pam.d/common-account***, ***/etc/pam.d/common-session*** y ***/etc/pam.d/common-password.*** El contenido de los ficheros es el siguiente:

```
# /etc/pam.d/common-auth
auth    [success=1 default=ignore]      pam_unix.so nullok_secure
auth    requisite                       pam_deny.so
auth    required                        pam_permit.so

# /etc/pam.d/common-account
account [success=1 new_authtok_reqd=done default=ignore]        pam_unix.so
account requisite                       pam_deny.so
account required                        pam_permit.so

# /etc/pam.d/common-session
session [default=1]                     pam_permit.so
session requisite                       pam_deny.so
session required                        pam_permit.so
session required        pam_unix.so

# /etc/pam.d/common-passwd
password        [success=1 default=ignore]      pam_unix.so obscure sha512
password        requisite                       pam_deny.so
password        required                        pam_permit.so
```

El significado de cada una de las líneas o agrupaciones por nombre de aparición es el siguiente:

- ***pam J'aildelay*** introduce un retraso cuando el ***login*** es incorrecto. En este caso el retraso es de ***3000000*** de microsegundos, expresado con el parámetro ***delay.*** Este módulo solo está disponible como tipo ***auth.***
- ***pamsecuretty*** se utiliza para que el usuario ***root*** solo pueda iniciar sesión desde las ***tty*** indicadas en el fichero ***/etc/securetty.*** Por otra parte, en lugar del control tradicional utiliza la nueva sintaxis. Entro otros se verifica si el usuario es desconocido o no para devolver fallo o éxito según convenga. Este módulo solo está disponible como tipo ***auth.***
- ***pam nologin deshabilita el*** acceso ***de*** cualquier usuario que no sea ***root*** siempre y cuando el fichero ***/etc/nologin*** exista. El contenido de ese fichero se mostrará al usuario en el momento del intento de acceso. Se ha utilizado como tipo ***auth.***
- ***pam selinux*** se establece con la finalidad de que PAM pueda funcionar en un entorno configurado con ***SELinux.*** Uno de sus parámetros de control verifica si ***SELinux*** está activado, devolviendo éxito en caso negativo. Este módulo solo está disponible como tipo ***session.***
- ***pam_env*** se utiliza para cargar variables de entorno. Por defecto se carga el fichero ***/etc/environment***, pero en el caso de Debian existe otra línea para cargar el fichero ***/etc/default/lócale.*** En este caso se ha utilizado como tipo ***session.***
- ***pam unix*** es el módulo encargado de la utilización del sistema de autenticación clásico de ***UNIX***, sea con sistema ***shadow***, ***/etc/shadow***, o sin él. En este caso se ha utilizado en todos los ficheros comunes, ***common-****, incluidos en el fichero ***/etc/pam.d/login.*** Dependiendo del tipo de uso, ya que puede ser de todos, se utilizarán unos parámetros u otros. Obviamente, funcionará de un modo u otro dependiendo del tipo de uso que se le esté dando, ya sea ***auth***, ***session***, etc.
- ***pam_permit*** y ***pam deny*** se utilizan para devolver éxito o fallo respectivamente. Se utilizan para asegurar de que ***PAM*** siempre devolverá algún tipo de valor. Se debe tener precaución con el uso de este módulo pues podría provocar fallos de seguridad si no se utiliza correctamente.
- ***pam_group*** es un módulo muy interesante, pues permite asignar determinados grupos a un usuario de forma dinámica dependiendo de cómo, dónde y cuándo haya iniciado sesión. Se apoya de la configuración del fichero ***/etc/security/group.conf.*** Se utiliza en modo ***auth*** y existe una amplia información de su utilización en ***man pam_group*** y ***man group.conf.*** Este módulo no tiene ningún efecto en principio ya que el fichero de configuración mencionado está completamente comentado.
- ***pamlimits*** permite establecer una serie de límites a nivel de sistema y usuario o grupo. La configuración para este módulo se realiza mediante el fichero ***/etc/security/limits.conf,*** cuyo contenido por defecto no influye en la configuración pues está completamente comentado. El módulo ***pam limits*** solo se utiliza en el nivel ***session.***
- ***pam lastlog***, ***pam motd*** y ***pam mail*** muestran información del último acceso realizado por el usuario y desde qué ubicación, el contenido del fichero ***/etc/motd*** y si el usuario

dispone de correo electrónico dentro de ***/var/mail/.*** Los tres módulos se han utilizado como tipo ***session.***

Login.defs

Este fichero, cuya ruta completa es /***etc/login.defs***, es el encargado de establecer ciertos parámetros por defecto para el sistema de autenticación clásico y estándar ***UNIX***, ***shadow.*** Muchos de sus parámetros han caído en desuso debido a la adopción de ***PAM*** como mecanismo de autenticación. Sin embargo, para tratar de fortificar la configuración del sistema existen algunos parámetros que pueden resultar interesantes. Los parámetros relevantes para su posterior modificación son los siguientes:

```
PASS_MAX_DAYS   99999         # Caducidad del password en días
PASS_WARN_AGE   7             # Aviso en días antes de la caducidad del password

LOGIN_RETRIES  5              # Intentos de login en caso de error
LOGIN_TIMEOUT  60             # Timeout en la espera de autenticación
```

Con los dos primeros parámetros del fichero ***/etc/login.defs*** se están generando unas entradas en el fichero ***/etc/shadow*** como las siguientes:

```
juanmi:$6$ift5BdKE$fP5rj..2sRzdLdVLIcg3Na40KlN4RQ.

DlQUQRGAotFKgSjonIXDmozzhltaP6FZAR3TxS/lXlHl3gYj2tPt680:15748:0:99999:7:::
```

Los campos con valor ***99999*** y 7 corresponden respectivamente con los valores por defecto mencionados del fichero ***/etc/login.defs.*** Para más información acerca del formato de este fichero se puede consultar el ***man shadow.***

Fortificando la configuración

Antes de empezar a modificar las configuraciones es necesario establecer qué es lo que se necesita modificar, atendiendo a los módulos de ***PAM*** y los ficheros de configuración implicados para ello. Como se hizo con ***sudo***, se utilizarán algunos de los usuarios definidos anteriormente para realizar el ejemplo de configuración.

Característica	Usuario	Módulo o fichero implicado
Los *passwords* caducarán en 30 días y se avisará 7 días antes del cambio.	Todos	La configuración se realiza en el fichero */etc/login.defs* y afecta a */etc/shadow*.
Los usuarios con *passwords* en blanco no podrán iniciar sesión de ninguna manera.	Todos	Modificar un parámetro de *pam_unix* utilizado como *auth* en el fichero *common-auth*.

Característica	Usuario	Módulo o fichero implicado
Cambios de *passwords* más seguros.	Todos	Se configurará el fichero *common-passwd* utilizando parámetros en *pam_unix*, además de *pam_cracklib*.
El inicio de sesión utilizando *login* solo permitirá un intento de *login*, estableciendo un *timeout* de 10 segundos.	Todos	La configuración se realiza en el fichero */etc/login.defs*.
Bloqueo de usuario si excede en 5 veces el número de intentos fallidos de sesión. El bloqueo se retirará una vez que transcurran 10 minutos.	Todos	La configuración se realiza mediante el módulo *pam_tally2* en el fichero */etc/pam.d/login*.
Como máximo un *login* simultaneo para el usuario.	*juanmi*	La configuración se realiza en el fichero */etc/security/limits.conf* que es utilizado por el módulo *pam_limits*.
Modificar la máscara por defecto aplicada a los ficheros.	Todos	La configuración se realiza en el fichero */etc/login.defs*, además de utilizar el módulo *pam_umask*.

Fichero /etc/login.defs

Atendiendo a las características indicadas en la tabla anterior se modificarán los siguientes parámetros con los valores especificados.

```
...
PASS_MAX_DAYS   30    # Caducidad del password en días
PASS_WARN_AGE   5     # Aviso de caducidad de password en días

UMASK           077   # Máscara por defecto para creación de ficheros
LOGIN_RETRIES   1     # Número máximo de intentos de login
LOGIN_TIMEOUT   10    # Timeout en la pantalla de login. Expresado en segundos
...
```

Fichero /etc/security/limits.conf

Para conseguir que el usuario ***juanmi*** solo pueda realizar un inicio simultáneo desde la pantalla de ***login*** debe establecerse la siguiente línea en el fichero indicado.

```
...
juanmi  -       maxlogins       1
```

Como se comentó anteriormente, el fichero está totalmente comentado por defecto, por lo que ayuda a la hora de establecer parámetros. A pesar de ello, la sintaxis que utiliza es:

```
<dominio>      <tipo> <objeto>       <valor>
```

- En ***dominio*** puede establecerse el valor para un usuario o grupo, siendo posible el uso de comodines para hacer que la configuración sea global.
- Para el campo ***tipo*** pueden establecerse los valores ***soft***, ***hard*** o -. Especifica el tipo de límite, siendo la última opción válida para especificar que el límite es de los dos tipos.
- Por ***objeto*** y ***valor*** se indica el límite en cuestión y el valor para dicho límite. Además del objeto ***maxlogins***, se podría haber utilizado por ejemplo ***cpu*** para limitar el tiempo de CPU disponible para el usuario.

Instalación de Cracklib y fichero /etc/pam.d/common-password

El módulo ***pam cracklib*** no está instalado por defecto en el sistema, por lo que será necesario instalarlo mediante los repositorios. En el caso de Debian el paquete a instalar es ***libpam-cracklib.*** Una vez instalado se modifica de forma automática el fichero ***/etc/pam.d/common-password***, por lo que el módulo ***pam cracklib*** empieza a utilizarse de inmediato. El aspecto del fichero es el siguiente:

```
password   requisite   pam_cracklib.so retry=3 minlen=8 difok=3
password    [success=1 default=ignore]    pam_unix.so obscure use_authtok try_
first_pass sha512
```

Tal y como está la configuración en estos instantes, ***cracklib*** requerirá de un ***password*** con una longitud mínima de 8 caracteres y que no existan 3 caracteres en el ***password*** anterior. Adicionalmente por defecto***, pam_cracklib*** no permite que se utilicen palíndromos, mismo ***password*** intercambiando mayúsculas y minúsculas, similares, ***passwords*** anteriores intercambiando letras y que no contenga el nombre de usuario entre otros.

En el caso del módulo***pam unix*** se agregado la opción ***try Jirst_pass,*** que junto con ***usejxuthok*** hará que este módulo utilice el ***password*** proporcionado a ***pamcracklib.*** De este modo no se solicitará nuevamente la introducción de un ***password.*** Por otra parte, el parámetro ***obscure*** también obligaba a ***pamjunix*** a que el ***password*** no fuera débil, pero ***pam_cracklib*** aporta más fortaleza y es posible añadirle más opciones.

Adicionalmente, se modificará la configuración anterior para que el usuario no pueda utilizar ***passwords*** ya utilizados, al menos durante 5 cambios. Para ello se ha de modificar la línea de ***pam_ unix*** agregando el parámetro ***remember=5***, que hará que los hashes de los ***passwords*** antiguos se almacenen en el fichero ***/etc/security/opasswd.*** La configuración por lo tanto quedaría del siguiente modo.

```
password   requisite   pam_cracklib.so retry=3 minlen=8 difok=3
password    [success=1 default=ignore]    pam_unix.so obscure use_authtok try_
first_pass sha512 remember=5
```

Fichero /etc/pam.d/common-session

Para que la aplicación de la máscara establecida en el fichero ***/etc/login.defs*** sea efectiva, es necesaria la utilización del módulo ***pam umask.*** La línea a agregar sería la siguiente:

```
session optional pam_umask.so
```

El valor establecido en el fichero ***/etc/login.defs*** se utiliza para los permisos aplicados al directorio ***HOME***, en la creación de un usuario con ***useradd*** utilizando el parámetro ***-m.***

A partir de que este momento los ficheros creados tendrán unos permisos del tipo ***600*** por defecto, lo cual aumenta la seguridad evitando que otros usuarios puedan leer el contenido. Si ya existían usuarios creados es muy recomendable modificar los permisos de sus directorios ***HOME*** a ***700***, ya que por defecto en Debían se establecen a ***755.***

Fichero /etc/common-auth

En este fichero se modificará el módulo ***pamunix*** para evitar que los usuarios sin ***password*** establecido puedan acceder al sistema, ni siquiera por las ***securetty*** establecidas. Por otra parte, se establecerá la configuración para utilizar ***pam_tally2*** y bloquear usuarios con intentos de inicio de sesión erróneos.

```
auth     required       pam_tally2.so deny=5 unlock_time=600
auth    [success=1 default=ignore]      pam_unix.so
...
```

Al módulo ***pam_unix*** se le ha eliminado la opción ***nullok secure.*** Con respecto a ***pam_tally2***, existe una aplicación adicional llamada ***pam_tally2*** que ayudará a gestionar los contadores de los usuarios afectados por el módulo ***pam_tally2.***

Efectos de la configuración

Algunos de los resultados obtenidos con la configuración anterior son los siguientes.

```
juanmi:$6$LJ5co0cX$.jHqMCJQq23yALUv5n1FmEjQz9U.pWa5qUHbYgnct8SF70sb6G2eU.
bQMiGzXq8cGAECujLIrgcoCn6/HGFA71:15753::30:5:::
```

Se trata de la línea del fichero ***/etc/shadow*** para el usuario ***juanmi.*** Se puede observar cómo se ha establecido el valor ***30 y 5*** para la caducidad del ***password*** y el aviso de modificación.

```
squeetest login: juanmi
Password:
Too many logins for 'juanmi'.

Permission denied
```

Fig 5.01: Intento de segundo *login* para el usuario *juanmi*. Ocasionado por *pam_limit.*

Con ***pam_tally2*** se bloqueaba a los usuarios que trataban de acceder al sistema más de un número establecido de veces de forma errónea. Alguien ha tratado de acceder con la cuenta de usuario ***pablo*** introduciendo un***password*** erróneo. El resultado es el siguiente:

```
squeetest nombre: pablo
La cuenta está bloqueada debido a 8 logueo fallidos
```

Puede consultarse el contador de errores de ***pablo*** utilizando la herramienta ***pam_tally2***.

```
squeetest:~# pam_tally2
Login           Failures Latest failure     From
pablo                8    02/17/13 16:20:51
```

Para finalizar con los resultados, el usuario ***german*** tratará de modificar su ***password*** utilizando el comando ***passwd.*** Tanto ***pam cracklib***, así como el parámetro ***remember*** de ***pam unix*** evitarán que pueda establecer un ***password*** débil o ya utilizado recientemente.

```
german@squeetest:~$ passwd german
Cambiando la contraseña de german.
(actual) contraseña de UNIX:
Contraseña:
Vuelva a escribir la nueva contraseña:
La contraseña ya se ha utilizado. Seleccione otra.
passwd: Error de manipulación del testigo de autenticación
passwd: password unchanged
german@squeetest:~$ passwd german
Cambiando la contraseña de german.
(actual) contraseña de UNIX:
Contraseña:
CONTRASEÑA INCORRECTA: es demasiado similar a la antigua
Contraseña:
CONTRASEÑA INCORRECTA: Es demasiado simple/sistemática.
Contraseña:
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
```

Las configuraciones realizadas son solo algunos de los ejemplos de lo que se puede realizar con ***PAM.*** Existen una gran variedad de módulos y opciones, que combinados entre sí pueden proporcionar una muy buena protección para la autenticación de un sistema.

Cuotas de almacenamiento

El espacio de un disco siempre es un aspecto que preocupa a un administrador de sistemas. Constantemente los usuarios o servicios llenan el sistema de ficheros, ya sea haciendo un uso indebido o normal. Cuando un disco o partición llega al máximo de su capacidad el sistema, dependiendo de la situación, dejará de responder y prestar sus servicios si no se libera espacio. Un ejemplo podría ser una partición donde se almacene el correo de los usuarios en la que no se controle el espacio. Es posible que un solo usuario, cargando datos adjuntos muy pesados, etc. pueda llegar a saturar el sistema de ficheros.

Para prevenir estas situaciones se utilizan las llamadas cuotas, que no son otra cosa que el establecimiento de límites de almacenamiento para usuarios o grupos. Las cuotas trabajan a nivel de sistema de ficheros, por lo que afectarán a una partición al completo y no a directorios; aunque siempre se puede jugar un poco con las particiones o con sistemas de ficheros virtuales donde montar directorios específicos.

Preparando el sistema

En el caso de Debían, como casi en la mayoría de distribuciones ***Linux***, el soporte de cuotas está desactivado por defecto. En este caso se va a habilitar la ***quota*** con ***journaling,*** así se evitarán problemas en caso de apagados inesperados de la máquina. La primera acción a realizar es la instalación de dos paquetes necesarios para el establecimiento de las cuotas.

```
# apt-get install quota quotatool
```

Como se comentaba, el establecimiento de cuota es una funcionalidad a nivel de sistema de ficheros por lo que habrá que comenzar por configurar el montaje de particiones para activar el soporte cuotas. Para ello hay que editar el fichero ***/etc/fstab*** y agregar las opciones ***usrjquota*** y/o ***grpjquota,*** además de ***jqfmt*** en la línea correspondiente de la partición deseada. En esta ocasión las cuotas se establecerán para la partición home.

```
UUID=b71c9c81-55b6-4ef3-92ec-fd5cbf8f189c   /home   ext4        errors=remount-
ro,usrjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0     0       2
```

Una vez remontada la partición ***/home*** aparecerá la siguiente información en el listado de particiones montadas con el comando ***mount.***

```
/dev/sdb1 on /home type ext4 (rw,errors=remount-
ro,usrjquota=aquota.user,grpjquota=aquota.group,jqfmt=vfsv0)
```

De momento todo es correcto, pero aún es necesario cargar el módulo necesario en el ***kernel*** para que el sistema pueda utilizar la funcionalidad de cuotas. Para activarlo y hacer que se cargue en el arranque es necesario realizar lo siguiente.

```
# modprobe quota_v2
# echo quota_v2 >> /etc/modules
```

Ya casi está todo listo, pero antes de establecer las cuotas es necesario generar un par de ficheros en la raíz del sistema de ficheros en el que se desean controlar las cuotas. Para ello se puede utilizar el comando ***touch*** para generar dos ficheros planos o bien utilizando el comando ***quotacheck*** c

```
# quotacheck -ugc /home
```

Acto seguido es necesario reiniciar el servicio ***quota***, que entre otras cosas inicializará los ficheros generados anteriormente que son los que contienen la información de las cuotas de los usuarios.

```
# service quota restart
Turning off quotas...done.
Checking quotas...done.
Turning on quotas...done.
```

Estableciendo la cuota de un usuario

Existen varias aplicaciones para establecer las cuotas de un usuario. En este caso se opta por la aplicación ***quotatool*** por su facilidad de uso.

Antes de continuar es necesario saber que al establecer la cuota para un usuario se indica un límite ***soft*** y otro ***hard.*** El último de ellos es el límite que el usuario nunca podrá rebasar, en cambio el ***soft*** será un límite superable por el usuario pero recibirá avisos por ello.

En este caso se establece la cuota para el usuario ***manu*** con un límite ***soft*** de ***450MiB*** y como ***hard 500MÍB.***

```
# quotatool -u manu -bq 450M -l 500M /home
```

Para comprobar que se ha establecido la cuota es posible utilizar el comando ***quota*** del siguiente modo.

```
# quota -u manu -vs
Disk quotas for user manu (uid 1003):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
      /dev/sdb1      0K    450M    500M               0       0       0
```

Otro método para modificar la cuota del usuario consiste en utilizar la aplicación ***edquota***, que permite la modificación de la cuota de un usuario de forma interactiva.

Sea como fuere, ahora que el usuario ***manu*** tiene establecida una cuota de ***500MiB*** se realizará una prueba para comprobar si efectivamente está funcionando. En primer lugar se creará un fichero de ***449MiB*** y luego se crearán ficheros adicionales para ver qué sucede.

```
manu@squeetest:~$ dd if=/dev/zero bs=1M count=449 of=./test1
manu@squeetest:~$ dd if=/dev/zero bs=1M count=1 of=./test2
sdb1: warning, user block quota exceeded.
manu@squeetest:~$ ls -lh
total 450M
-rw------- 1 manu manu 449M feb 17 21:19 test1
-rw------- 1 manu manu 1,0M feb 17 21:20 test2
manu@squeetest:~$ quota -u manu -vs
Disk quotas for user manu (uid 1003):
     Filesystem   space   quota   limit   grace   files   quota   limit   grace
      /dev/sdb1    451M*   450M    500M   6days       3       0       0
```

Como se puede observar el usuario manu ha rebasado el límite ***soft*** pero aún puede seguir utilizando más almacenamiento. Como última parte de este ejemplo se va a intentar escribir un fichero de ***500MB*** adicional a los que ya existen.

```
manu@squeetest:~$ dd if=/dev/zero bs=1M count=500 of=./test3
sdb1: write failed, user block limit reached.
dd: escribiendo «./test3»: Se ha excedido la cuota de disco
50+0 records in
49+0 records out
52420608 bytes (52 MB) copied, 0,0442609 s, 1,2 GB/s
manu@squeetest:~$ ls -lh
total 500M
-rw------- 1 manu manu 449M feb 17 21:19 test1
-rw------- 1 manu manu 1,0M feb 17 21:20 test2
-rw------- 1 manu manu  50M feb 17 21:24 test3
```

El comando ***dd*** tan solo ha podido escribir ***50MiB*** más, dado que era el máximo hasta llegar al límite ***hard*** del usuario ***manu.*** Se puede concluir por lo tanto que el sistema de cuotas está funcionando correctamente.

Por otra parte, es posible activar un aviso diario del estado de las cuotas, utilizando la aplicación ***warnquoía.*** Lo único que hay que realizar es establecer el valor ***runwarnquota*** a ***true*** en el fichero

/etc/default/quota. Eso hará que se ejecute la tarea programada en el ***cron.daily.*** Una vez que se ejecute, al usuario ***manu*** le llegaría un e-mail con el asunto "***Over quotcT.***

```
manu@squeetest:~$ mail
Mail version 8.1.2 01/15/2001.  Type ? for help.
"/var/mail/manu": 1 message 1 new
>N  1 root@localhost     Sun Feb 17 21:33   41/1373  Over quota
```

Es posible modificar las opciones para ***warnquota*** en el fichero ***/etc/warnquota.conf***

Monit

Si bien anteriormente se había hablado de ***Icinga*** para monitorizar las máquinas de un entorno de red, ahora llega el tumo de ***Monit.*** Se trata de una utilidad para monitorizar una gran variedad de eventos en sistemas ***Linux*** y realizar acciones en situaciones de fallo o límites que se establezcan en la configuración. Es decir, no solo se informa de que está ocurriendo un error o un servicio está saturado, sino que se puede tratar de solventar el problema de forma automática reiniciando el servicio, ejecutando algún script, etc.

Los eventos que puede monitorizar ***Monit*** son los siguientes:

- Estado general del sistema. Utilización de CPU, memoria y carga media entre otros.
- Aplicaciones o demonios ejecutándose en el sistema. ***Monit*** se encargará de que las aplicaciones configuradas para su monitorización respondan en todo momento. Se alertará en caso de sobrecarga o detención, además de realizar una actuación al respecto. Un ejemplo típico de esto sería la monitorización del servidor web ***Apache***, tratando de alertar sobre un exceso de peticiones y elevado consumo de recursos.
- Monitorización de ficheros. ***Monit*** puede detectar cambios en los ***timestamps*** de ficheros, ***hashes*** de los mismos o modificaciones de tamaño. Sin duda resulta muy útil para ficheros sensibles que no deberían ser modificados salvo excepciones.
- Conexiones de red ***TCP***, ***UDP*** y ***sockets Unix.*** Además es capaz de manejar protocolos como ***HTTP***, ***SMTP o SIP*** entre otros.
- Otra opción destacable reside en la posibilidad de ejecutar ***scripts*** o aplicaciones cada cierto tiempo. Puede parecer que para esto ya existe ***cron***, pero sin embargo utilizando ***Monit*** es posible controlar la salida de la aplicación a modo de depuración y testeo.

Instalación y estructura

Como ocurre con la mayoría de las aplicaciones, ***Monit*** puede instalarse compilando sus fuentes o bien desde los repositorios de las distribuciones. En este caso para Debían se opta por la instalación desde los repositorios. Para ello se realiza algo como:

```
# apt-get install monit
```

En el momento en que ***apt*** descargue los ***509 kiB*** necesarios para instalar ***Monit***, el sistema generará los siguientes ficheros relevantes para la puesta en marcha y configuración de la aplicación.

- ***/usr/sbin/monit.*** Es el binario en sí. Podrá ejecutarse tanto en modo demonio o ***background***, como en primer plano o bien actuando de cliente para controlar una instancia de ***Monit*** ejecutándose como demonio. El arranque automático y en segundo plano de la aplicación está controlado por el script de inicio ***/etc/init.d/monit.***
- ***/etc/monit/monitrc.*** Es el fichero de control de ***Monit.*** En él se realiza toda la configuración, la monitorización de servicios, etc. Es la forma más común de configurarlo y para ello se utiliza una sintaxis de muy fácil comprensión, por lo que al leer las líneas de configuración se conocerá inmediatamente su cometido. El fichero de configuración por su parte está dividido en tres secciones.
  - Opciones globales, establecidas con la palabra reservada ***set.***
  - Entradas ***inelude.*** Indica configuración adicional de otros ficheros. Por defecto existe una entrada de este tipo para los ficheros del directorio ***/etc/monit/conf.d.***
  - Configuraciones de servicios. Estas entradas comienzan por ***check.***
- ***/etc/default/monit.*** Se trata de un fichero de configuración básico para el funcionamiento de ***Monit.*** Es necesario establecer el parámetro ***startup*** como ***1*** para que ***Monit*** pueda funcionar.

Ejemplo de configuración

Monit dispone de un amplio manual, ***man monit*** donde se especifican todos los parámetros de configuración y ejemplos; así como la explicación de todas sus directivas de configuración. La idea es que con el siguiente ejemplo se entienda el funcionamiento de ***Monit*** para que sirva como base en otras configuraciones.

Lo primero de todo consiste en detallar qué es lo que se necesita monitorizar con ***Monit.*** A continuación se exponen los puntos clave para la configuración.

- ***Monit*** realizará una comprobación de los servicios cada minuto, ***60*** segundos. Es decir, un ciclo equivaldrá a ***60*** segundos.
- Se informará de excesiva carga en el sistema a través de correo, utilizando la dirección definida globalmente.
- Se monitorizará el servicio ***SSH.*** En principio solo se contratará que el servicio siempre esté activo.
- Para el servidor web Apache se contralará su uso de ***CPU*** y si el servicio está activo. En caso de que el servicio consuma excesivos recursos. Se notificará por correo de este incidente.
- Se monitorizará el fichero ***/etc/resolv. conf*** en busca de modificaciones. En caso de haberlas se notificará por correo.

El contenido del fichero de configuración será parecido al siguiente.

```
##########################
```

```
## Generales
#########################
# Chequeos cada minuto
set daemon  60
# Servidor para la entrega de correo
set mailserver localhost
# Buzón general para las alertas
set alert root@localhost
# Almacenar 100 eventos como máximo si falla el MTA
set eventqueue
     basedir /var/monit
     slots 100
# Fichero para el log
set logfile syslog

#########################
## Servicios
#########################
# Carga del sistema
check system localhost
        if loadavg (1min) > 4 then alert
        if loadavg (5min) > 2 then alert

# Cambios en /etc/resolv.conf
check file resolv.conf path /etc/resolv.conf
        if changed checksum then alert

# Verificar si SSH está activo
check process sshd with pidfile /var/run/sshd.pid
        start program = "/etc/init.d/ssh start"
        stop program = "/etc/init.d/ssh stop"
        if failed host localhost
                port 22 protocol ssh
        then start
```

```
# Reinicio de Apache bajo mucha carga
check process apache with pidfile /var/run/apache2.pid
        start program = "/etc/init.d/apache2 start"
        stop program  = "/etc/init.d/apache2 stop"
        if cpu > 40% for 2 cycles then alert
        if cpu > 80% for 5 cycles then restart
```

En el log del sistema ***/var/log/syslog,*** bajo una situación de carga, se podrían encontrar los siguientes mensajes.

```
...
Feb 19 00:24:11 squeetest monit[4843]: 'localhost' loadavg(5min) of 6.7 matches
resource limit [loadavg(5min)>2.0]
Feb 19 00:24:11 squeetest monit[4843]: 'localhost' loadavg(1min) of 9.0 matches
resource limit [loadavg(1min)>4.0]
```

Por otra parte, al correo del administrador habrán llegado mensajes como los siguientes, en los que se indica que el fichero ***/etc/resolv.conf*** ha sido modificado y que el servicio ***SSH*** no está disponible.

```
N 13 monit@squeetest     Tue Feb 19 00:33    30/859    monit alert --  Checksum
changed resolv.conf
N 14 monit@squeetest     Tue Feb 19 00:33    30/824    monit alert --  Does not
exist sshd
```

Lo mostrado es una configuración base, pero ***Monit*** ofrece una amplia variedad de parámetros para su configuración, que unido a configuraciones de correo, diferentes tipos de notificaciones, lanzamiento de Scripts personalizados, etc. lo hace una herramienta muy poderosa y útil para el administrador de sistemas. Es muy recomendable su utilización en prácticamente cualquier tipo de entorno.

5. Port-Knocking

Port-Knocking o golpeo de puertos es un mecanismo que permite crear una capa de seguridad por oscuridad, es decir ayuda a proteger un recurso de un sistema mediante su ocultación. La técnica de ***Port-Knocking*** se utiliza para permitir el acceso remoto bajo demanda. Un ejemplo muy común para implementaciones de esta solución es utilizarlo en conjunción con un servidor ***SSH***, ***Secure SHell*** que será estudiado en profundidad en el siguiente capítulo.

Continuando con el ejemplo de ***SSH*** se supone un entorno que lo implementa. Para tratar de securizar dicho entorno, se desea evitar que escaneos de red, con ***nmap*** por ejemplo, desvelen que esa máquina dispone de un servicio ***SSH.*** La idea consiste en que inicialmente el puerto utilizado por ***SSH*** esté bloqueado mediante reglas de ***firewall*** y solo cuando un usuario desee acceder se le conceda el acceso exclusivamente a él. Para conseguir esto se utiliza ***Port-Knocking***, que consiste en que una máquina ***firewall***, que ejecuta el proceso de gestión de ***Port-Knocking***, analice los paquetes que le llegan por su interfaz de red. En caso de detección de una determinada secuencia de paquetes ***SYN*** a varios puertos, el demonio de ***Port-Knocking*** se encargará de establecer reglas en el ***firewall*** para permitir el acceso del cliente que inició la secuencia. Acto seguido el cliente tendría acceso ***SSH*** a la máquina. El proceso de configuración y establecimiento de sesión puede resumirse en los siguientes puntos.

1. Proceso a proteger ejecutándose en la máquina servidor. Máquina^.
2. La máquina ***A*** ejecuta un ***firewall*** que bloquea todas las conexiones entrantes nuevas. El servicio SSH por lo tanto es inaccesible e indetectable.
3. La máquina ***A*** tiene configurado un demonio de ***Port-Knocking.*** La configuración consiste en abrir el puerto para ***SSH*** en caso de recibir una secuencia de paquetes ***SYN*** a los puertos ***1234***, ***4321*** y ***9876*** por ejemplo. Esa secuencia debe realizarse en menos de una ventana de tiempo preconfigurada en el demonio de ***Port-Knocking***, por ejemplo ***5*** segundos.
4. La máquina ***B*** o cliente que desea conectarse al servidor ***SSH*** no dispone de acceso a la máquina ***A*** por el bloqueo realizado.
5. La máquina ***B*** ejecuta un cliente de ***Port-Knocking*** y envía al servidor la secuencia establecida en la configuración.
6. La máquina ***A*** detecta que la secuencia es correcta y habilita la conexión entrante desde la máquina ***B*** hacia ***A*** para ***SSH.***
7. La máquina ***B*** puede conectar mediante ***SSH*** a la máquina ***A.***

Cabe destacar que para el ejemplo, la máquina que ejecuta el servidor ***SSH.***, el ***firewall*** y el servicio de ***Port-Knocking*** son la misma. No es obligatoria esa topología, ya que el servicio ***Port-Knocking*** puede agregar al ***firewall*** reglas de tipo ***FORWARD*** para paquetes destinados a otras máquinas. Obviamente, el servicio ***Port-Knocking*** siempre deberá ejecutarse en una máquina que tenga configuradas reglas de ***firewall.***

SPA, Single Packet Authorization

A pesar de que ***Port-Knocking*** puede resultar útil a priori, es un concepto demasiado básico y si algún atacante adquiere la secuencia de paquetes utilizada podrá terminar habilitando un acceso al servidor ***SSH.*** Por supuesto, después tendría que lidiar con la seguridad del servicio ***SSH*** pero esta primera capa la habría roto fácilmente.

Para evitar la situación anterior e incluso añadir más opciones y seguridad, se creó el ***SPA*** o ***Single Packet Authorization.*** Puede considerarse como una variante de ***Port-Knocking*** pero sin embargo no

se envía una secuencia de paquetes sino un solo único paquete con información cifrada. Esto añade privacidad y autenticación. Adicionalmente agrega determinados parámetros que impedirán utilizar los paquetes capturados por un atacante para un desplegar un ataque ***replay*** y conseguir el acceso.

fwknop. Instalación y configuración

La implementación de ***SPA*** utilizada es ***fwknop***, ***Pire Wall KNock OPerator***, cuya instalación puede realizarse mediante los repositorios oficiales, diferenciándose los paquetes para servidor y cliente. Entre otras muchas características, ***fwknop*** ofrece las siguientes funcionalidades.

- Diferentes modos de funcionamiento. Es posible configurar ***fwknop*** para funcionar utilizando ***SPA***, ***Port-Knocking*** tradicional o mediante el envió por parte del cliente de un fichero ***pcap*** previamente configurado en el servidor ***fwknop.***
- Modos de cifrado para ***SPA.*** Por defecto se utiliza cifrado simétrico mediante el algoritmo ***Rijndael***, AES. Sin embargo es posible utilizar infraestructura de clave pública con ***GNUPG.***
- Posibilidad de creación de reglas de***firewall*** avanzadas. Permite crear reglas ***FORWARD***, ***OUTPUT\ DNAT***, ***SNAT***, etc. en el caso de ***iptables.***
- Existe la opción de configurar tanto el servidor como el cliente para que el intercambio del paquete cifrado ***SPA*** se realice utilizando ***HTTP.***

Para la configuración de ejemplo se seguirá el mismo esquema que en el ejemplo inicial.

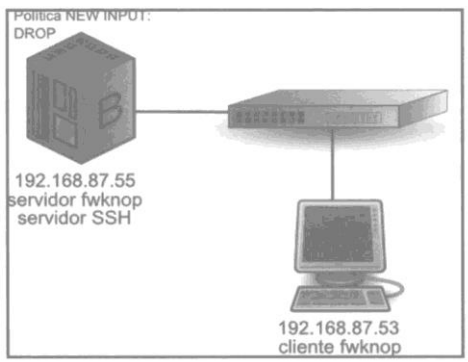


Fig 5.02: Entorno de ejemplo para *Port-Knocking*.

Configuración del servidor fwknop. Máquina A

Esta máquina dispone de servicio ***SSH*** y de un ***firewall*** que desecha todas las conexiones entrantes nuevas, por lo que el acceso ***SSH*** inicial es imposible. La configuración del ***firewall*** es la siguiente.

```
iptables -F
iptables -P INPUT DROP
iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -i eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Para disponer del servicio ***fwknop*** es necesario instalar el paquete ***fwknop-server.*** Una vez instalado se dispondrán de los siguientes archivos relevantes.

- ***/etc/fwknop/fwknop.conf.*** Es el fichero de configuración principal e ***fwknop.*** En él se definen valores muy variados como el correo de contacto o el tipo de ***firewall*** a usar para habilitar las conexiones. El fichero de ejemplo está muy detallado, comentando cada una de las posibilidades.
- ***/etc/fwknop/access.conf.*** Permite la configuración de los puertos o ***sockets*** a los que permitir el acceso en caso de recibir una petición correcta.
- ***/etc/init.d/fwknop.*** Es el ***script*** de inicio de ***fwknop.*** Antes de poder utilizar el demonio es necesario editar el fichero ***/etc/default/fwknop-server*** y modificar el valor del parámetro ***STAR TDAEMON.***
- ***/usr/sbin/knopmd***, ***/usr/sbin/knoptm***, ***/usr/sbin/knopwatchd.*** Son los binarios auxiliares utilizados por ***fwknop.*** El primero de ellos se encarga de tratar los ***logs*** de ***iptables*** en caso de utilizar el ***Port-Knocking*** tradicional. El segundo se utiliza para mantener las reglas del ***firewall***; por ejemplo cuando las elimina por ***timeouts.*** El último binario se encarga de que ***fwknop*** esté siempre operativo atendiendo a su estado de ejecución.

La configuración escogida en este caso es la siguiente, mostrando primero el fichero de configuración ***/etc/fwknop/access. conf.***

```
SOURCE: ANY;                          # Cualquier origen puede solicitarlo
OPEN_PORTS: tcp/22;                   # Puerto para SSH
KEY: S7p3rS3cr3t_P4ssW0rd;            # El password simétrico para el cifrado
FW_ACCESS_TIMEOUT: 30;                # Elim. regla en caso de inactiv. de 30 seg
```

Por otra parte, el fichero ***/etc/fwknop/fwknop.conf*** es válido en su configuración por defecto, siendo necesaria la modificación de los parámetros ***EMAIL ADORES SES*** y ***HOSTNAME.*** A pesar de ello, algunos de los parámetros importantes son los siguientes.

```
EMAIL_ADRESSES        root@localhost;
HOSTNAME              test;
FIREWALL_TYPE         iptables;      # Sistema de firewall utilizado.
AUTH_MODE             PCAP;          # ULOG_PCAP para Port-Knocking.
PCAP_INTF             eth0;          # Interfaz de red utilizada.
ENABLE_PCAP_PROMISC   Y;             # Habilitar modo promiscuo.
ENABLE_SPA_PACKET_AGING      Y;      # Se utiliza el Timestamp. Evita replay.
MAX_SPA_PACKET_AGE    120;           # Edad máxima de un paquete en segundos.
...
```

En este instante el servicio ***fwknop*** está preparado para recibir peticiones y abrir el puerto de ***SSH*** para los clientes que lo precisen. Las reglas ***iptables*** están aplicadas y su aspecto es el siguiente.

```
root@test:~# iptables -nvL
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0

    0     0 ACCEPT     all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain FWKNOP_INPUT (0 references)
 pkts bytes target     prot opt in     out     source               destination
```

Fig 5.03: Reglas *iptables*.

Como se observa en la imagen anterior, ***fwknop*** ha creado una cadena llamada ***FWKNOP INPUT*** en la tabla ***FILTER*** de ***iptables.*** Ahí es donde se agregarán las reglas de forma automática para este ejemplo. En el servidor por lo tanto está todo operativo para comenzar a funcionar con el entorno de ***Port-Knocking***, por lo que llega el tumo de conectar desde el cliente.

Utilizando fwknop para habilitar el acceso remoto. Máquina B

Inicialmente, si desde esta máquina se intenta acceder vía ***SSH*** a la máquina ***A*** se obtendrá el siguiente resultado.

```
squeetest:~# ssh root@192.168.87.55
ssh: connect to host 192.168.87.55 port 22: Connection timed out
```

Al igual que se realizó en la máquina ***A,*** es necesario instalar ***fwknop*** pero esta vez el cliente. En el caso de Debían se encuentra en los repositorios oficiales instalando el paquete ***fwknop-client.*** Una vez instalado y a modo de que la máquina ***A*** abra su acceso a ***SSH*** para la máquina ***B*** se debe realizar lo siguiente.

```
squeetest:~# fwknop -A tcp/22 -D 192.168.87.55 -a 192.168.87.53

[+] Starting fwknop client (SPA mode)...
[+] Enter an encryption key. This key must match a key in the file
    /etc/fwknop/access.conf on the remote system.

Encryption Key:
```

```
[+] Building encrypted Single Packet Authorization (SPA) message...
[+] Packet fields:

        Random data:    8715289205583774
        Username:       root
        Timestamp:      1361389693
        Version:        1.9.12
        Type:           1   (access mode)
        Access:         192.168.87.53,tcp/22
        SHA256 digest:  HNfDrc2OlDhmDLmH0IOE7uTbzKj4j1amj/oQOSajFWE

[+] Sending 182 byte message to 192.168.87.55 over udp/62201...
```

Los parámetros del comando indican respectivamente el puerto a habilitar, la dirección IP del servidor ***fwknop*** y por último la IP que debe habilitarse. La aplicación dispone de muchos otros parámetros para modificar su comportamiento de acuerdo a las configuraciones establecidas. Para ello puede consultarse su ***man.***

Si el mensaje enviado ha cumplido con los requisitos establecidos en el servidor ***fwknop,*** el acceso estará habilitado para la máquina ***B*** hacia el puerto ***22/tcp*** de la máquina ***A.***

```
squeetest:~# ssh root@192.168.87.55
root@192.168.87.55's password:
```

En efecto, una vez que se lanza la conexión desde ***B*** hacia ***A*** utilizando el cliente ***SSH***, se solicita el ***password*** al usuario ***root*** para acceder de forma remota. Todo ha funcionado correctamente pero ¿qué ha pasado en la máquina ***Al***

Acceso habilitado. Máquina A

En la máquina^, atendiendo al ***log*** del sistema se comprueba que el acceso debe concederse para la máquina ***B*** pues se ha autenticado contra el servidor ***fwknop*** de forma correcta.

```
Feb 20 20:48:13 test fwknopd: received valid Rijndael encrypted packet from:
192.168.87.53, remote user: root, client version: 1.9.12 (SOURCE line num: 26)
Feb 20 20:48:13 test fwknopd: add FWKNOP_INPUT 192.168.87.53 -> 0.0.0.0/0(tcp/22)
ACCEPT rule 30 sec
```

Adicionalmente ***fwknop*** se ha encargado de crear una nueva regla en ***iptables*** para permitir el acceso de la máquina B***.***

```
Chain FWKNOP_INPUT (1 references)
 pkts bytes target     prot opt in     out     source               destination
   28  3657 ACCEPT     tcp  --  *      *       192.168.87.53        0.0.0.0/0
tcp dpt:22
```

Esa regla, que permite una conexión entrante nueva hacia el puerto ***22/tcp*** desde la dirección 1P ***192.168.87.53*** se eliminará a los ***30*** segundos de establecerse. Es el comportamiento que tiene por defecto ***fwknop.*** En este caso no influye a la hora del funcionamiento posterior de ***SSH*** pues el cortafuegos está configurado para aceptar las conexiones ***ESTABLISHED.***

6. Actualizaciones seguras en Debían

Sin lugar a dudas, uno de los aspectos en los que la distribución Debian destaca sobre las demás es el sistema de gestión de paquetes. El corazón para ello reside en la utilidad ***dpkg***, pero a la hora de mantener actualizado un sistema no es la solución más adecuada. Para este último cometido se desarrolló ***apt,*** un ***front-end*** de ***dpkg*** que permite la instalación, desinstalación y actualización de software del sistema ya pueda ser con paquetes de software locales o bien en repositorios remotos.

El sistema api, junto con la configuración de los repositorios remotos utilizados ayudará a mantener actualizado un sistema Debian. Para conseguir conectar con sistemas remotos que ofrecen software para Debian, la configuración de dichos repositorios se realiza en el fichero ***/etc/apt/sources.list*** o bien en ficheros segmentados dentro del directorio ***/etc/apt/sources.list.d.*** En cada uno de los ficheros estarán definidas direcciones a las que ***apt*** debe conectarse para obtener información de los nuevos paquetes, actualizaciones, etc. Un extracto del fichero ***sources.list*** podría ser la siguiente.

```
deb http://security.debian.org/ squeeze/updates main
deb-src http://security.debian.org/ squeeze/updates main
```

Debian por defecto está configurado con una serie de repositorios oficiales de la distribución, como los de seguridad del ejemplo de arriba. Sin embargo, el usuario siempre tiene la posibilidad de agregar nuevos repositorios con más software no oficial, réplicas alternativas, etc. Esto último, junto a algunos de los ataques desplegables en una red que se han visto anteriormente, puede conducir a situaciones peligrosas en las que se llegue a instalar software no deseado en una máquina. ¿Qué propone Debian ante este potencial problema?

¿Es seguro apt?

La respuesta es sí, siempre y cuando la persona encargada de la instalación de software y actualización del mismo preste la atención necesaria.

Inicialmente, ***apt*** verificaba la integridad de los paquetes descargados, ya que en el listado de paquetes, ficheros ***Packages*** y ***Packages.gz,*** están contenidos todos los paquetes disponibles en un repositorio junto con información del paquete, así como un ***hash*** de verificación. Lo que sucedía es que no había ningún método para verificar la procedencia de los paquetes. Posteriormente ***apt*** fue modificado para trabajar de forma más segura, firmando mediante ***GPG*** el contenido de unos ficheros llamados ***Pelease*** que son los que verifican a su vez la integridad de ficheros como el ***Packages.gz*** mencionado anteriormente.

Es una explicación demasiado condensada, pero en la ***Wiki*** de Debian existe una documentación muy interesante sobre ***SecureApt.*** También puede consultarse ***man apt-secure.*** La conclusión de todo lo anterior es que hoy en día Debian verifica mediante firmas ***GPG*** la procedencia de los paquetes.

Gestión de claves públicas para la verificación de firmas

Cuando se instala desde cero un sistema Debian, se configura ***apt*** para que haga uso de las claves públicas para la autenticación de los paquetes descargados desde los repositorios oficiales. Las claves públicas instaladas por defecto, cuyas parejas privadas pertenecen a Debian, se almacenan dentro del fichero ***/etc/apt/trust ed. gpg,*** que es el ***keyring*** o anillo de claves que utiliza ***apt.***

Para la gestión de las claves se utiliza ***apt-key*** junto a ***gpg***, que unido a una serie de parámetros permitirá exportar claves, eliminarlas, añadir nuevas, etc. Por ejemplo, realizar un listado de las claves públicas junto con sus huellas se realizaría del siguiente modo.

```
squeetest:~# apt-key finger
/etc/apt/trusted.gpg
--------------------
pub   1024D/F42584E6 2008-04-06 [caducó: 2012-05-15]
      Huella de clave = 7F5A 4445 4C72 4A65 CBCD  4FB1 4D27 0D06 F425 84E6
uid    Lenny Stable Release Key <debian-release@lists.debian.org>

pub   4096R/55BE302B 2009-01-27 [caducó: 2012-12-31]
      Huella de clave = 150C 8614 919D 8446 E01E  83AF 9AA3 8DCD 55BE 302B
uid    Debian Archive Automatic Signing Key (5.0/lenny) <ftpmaster@debian.org>

pub   2048R/6D849617 2009-01-24 [caducó: 2013-01-23]
      Huella de clave = F6CF DE30 6133 3CE2 A43F  DAF0 DFD9 9330 6D84 9617
uid    Debian-Volatile Archive Automatic Signing Key (5.0/lenny)

pub   4096R/B98321F9 2010-08-07 [caduca: 2017-08-05]
      Huella de clave = 0E4E DE2C 7F3E 1FC0 D033  800E 6448 1591 B983 21F9
```

```
uid    Squeeze Stable Release Key <debian-release@lists.debian.org>

pub   4096R/473041FA 2010-08-27 [caduca: 2018-03-05]
      Huella de clave = 9FED 2BCB DCD2 9CDF 7626  78CB AED4 B06F 4730 41FA
uid    Debian Archive Automatic Signing Key (6.0/squeeze) <ftpmaster@debian.org>

pub   4096R/46925553 2012-04-27 [caduca: 2020-04-25]
      Huella de clave = A1BD 8E9D 78F7 FE5C 3E65  D8AF 8B48 AD62 4692 5553
uid    Debian Archive Automatic Signing Key (7.0/wheezy) <ftpmaster@debian.org>

pub   4096R/65FFB764 2012-05-08 [caduca: 2019-05-07]
      Huella de clave = ED6D 6527 1AAC F0FF 15D1  2303 6FB2 A1C2 65FF B764
uid    Wheezy Stable Release Key <debian-release@lists.debian.org>
```

Para verificar qué ocurriría si no existiera la clave oficial de Debian ***Squeeze***, se procede a su eliminación del ***keyring*** del siguiente modo.

```
squeetest:~# apt-key del B98321F9
OK
squeetest:~# apt-get update
Des:1 http://ftp.de.debian.org squeeze Release.gpg [1672 B]
Ign http://ftp.de.debian.org squeeze/main Sources/DiffIndex
...
Obj http://ftp.de.debian.org squeeze/main Sources
Obj http://ftp.de.debian.org squeeze/main i386 Packages
Descargados 1672 B en 0seg. (2600 B/s)
Leyendo lista de paquetes... Hecho
W: No existe ninguna clave pública disponible para los siguientes identificadores
de clave:
64481591B98321F9
```

Como se puede observar se está obteniendo una alerta por parte de ***apt*** advirtiendo de que la firma que contiene el fichero ***Release***, mencionado anteriormente, no puede ser verificada localmente. Si se ignorase dicha advertencia se podría instalar paquetes con normalidad, pero no se estaría verificando la fuente.

Para agregar una nueva clave al repositorio de claves de ***apt***, utilizando como ejemplo la restauración de la clave pública de ***Squeeze***, se realizarían los siguientes pasos.

```
squeetest:~# gpg --recv-keys B98321F9
gpg: solicitando clave B98321F9 de hkp servidor keys.gnupg.net
gpg: clave B98321F9: clave pública "Squeeze Stable Release Key <debian-release@
lists.debian.org>" importada
gpg: 3 dudosa(s) necesarias, 1 completa(s) necesarias,
modelo de confianza PGP
gpg: nivel: 0  validez:   1  firmada:   0  confianza: 0-, 0q, 0n, 0m, 0f, 1u
gpg: siguiente comprobación de base de datos de confianza el: 2013-12-08
gpg: Cantidad total procesada: 1
gpg:               importadas: 1  (RSA: 1)
squeetest:~# gpg -a --export B98321F9 | apt-key add -
OK
```

Es decir, utilizando los servidores ***gpg*** disponibles se adquiere la clave pública de Debian ***Squeeze*** nuevamente y posteriormente se importa utilizando ***apt-key add*** -. Es posible obtener las claves de Debian desde la dirección http://ftp-master.debian.org/keys.html**.**

Actualizaciones y MitM

Puede que exista la posibilidad en la que un atacante suplante a un repositorio de software. Por ejemplo uno oficial de Debian. Para esta demostración se supone el siguiente entorno.

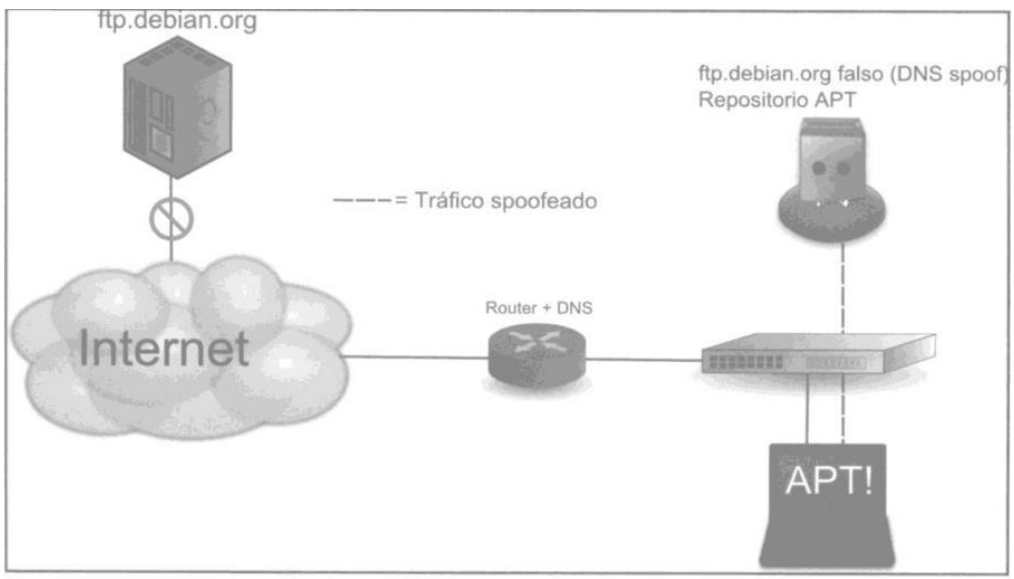


Fig 5.04: *MiTM* para suplantar repositorios *APT.*

Para este ejemplo, la máquina atacante tendrá un repositorio clonado con el paquete ***sudo*** modificado con la finalidad de realizar acciones perjudiciales para la víctima y conseguir información sensible. La dirección suplantada mediante ***DNS Spoofing*** es ftp.de.debian.org**.** El cliente continuará con sus repositorios configurados de forma correcta y este será el comportamiento a la hora de actualizar los repositorios y actualizar el software del sistema.k

```
squeetest:~# apt-get update
Ign http://ftp.de.debian.org squeeze Release.gpg
Ign http://ftp.de.debian.org/debian/ squeeze/main Translation-en
Ign http://ftp.de.debian.org/debian/ squeeze/main Translation-es
Des:1 http://ftp.de.debian.org squeeze Release [1528 B]
Des:2 http://ftp.de.debian.org squeeze/main Sources [20 B]
Des:3 http://ftp.de.debian.org squeeze/main i386 Packages [533 B]
Des:4 http://security.debian.org squeeze/updates Release.gpg [836 B]
Ign http://security.debian.org/ squeeze/updates/main Translation-en
Ign http://security.debian.org/ squeeze/updates/main Translation-es
Des:5 http://security.debian.org squeeze/updates Release [87,0 kB]
Des:6 http://security.debian.org squeeze/updates/main Sources [122 kB]
Des:7 http://security.debian.org squeeze/updates/main i386 Packages [368 kB]
Descargados 580 kB en 1seg. (358 kB/s)
Leyendo lista de paquetes... Hecho
```

De antemano resulta sospechoso que para el dominio ftp.de.debian.org no se haya descargado el fichero ***Release.gpg.*** Ese fichero es necesario para verificar la autenticidad y origen de los paquetes pues contiene la firma del fichero ***Release.*** Esto quiere decir que el atacante no dispone de un servidor que utilice clave pública para autenticar sus paquetes. En cambio, para el caso de ***security.debían, org*** sí se ha descargado el fichero ***Release.gpg*** porque esa dirección del repositorio de seguridad no está suplantada.

El atacante ha aumentado el número de versión para el paquete ***sudo*** suplantado, por lo que cuando la víctima realice una actualización aparecerá como actualización disponible. Es decir, en la máquina víctima aparecería lo siguiente.

```
squeetest:~# apt-get upgrade
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se actualizarán los siguientes paquetes:
  sudo
```

```
1 actualizados, 0 se instalarán, 0 para eliminar y 0 no actualizados.
Necesito descargar 593 kB de archivos.
Se utilizarán 0 B de espacio de disco adicional después de esta operación.
¿Desea continuar [S/n]?
AVISO: ¡No se han podido autenticar los siguientes paquetes!
  sudo
¿Instalar estos paquetes sin verificación [s/N]?
```

Como se puede observar se avisa de que el paquete no puede ser verificado. En ese instante la víctima debería intuir que podría estar siendo víctima de un ataque. Por defecto ***apt***, en caso de pulsar **<*Enter*>** no instalará el paquete. Si el usuario obviara la alerta podría suponer consecuencias desastrosas para el sistema y la información sensible almacenada en él. Todo ello dependiente del objetivo del atacante. A modo de prueba, para este ejemplo se modificó la descripción del paquete ***sudo*** para hacer más visual la modificación y suplantación del paquete.

```
squeetest:~# apt-cache show sudo
Package: sudo
Version: 1.7.5p4-2.squeeze.3
Architecture: i386
...
Size: 593132
SHA256: 1ec1032ff55011f7c20fb6f80d07649f556930eab6e1c3ea244a8b1daa9cc506
SHA1: 9b05340fcdf9be28c52bce0011e395209981b9f1
MD5sum: bc81b833096ded072d86c8ca409d613b
Description: Provide limited super user privileges to specific users
 Sudo is a program .... blablabla .... El libro de Hardening mola...
 This version is built with minimal shared library dependencies, use the
 ...
```

La conclusión a la que se puede llegar es que si el usuario controla en todo momento las claves públicas de los repositorios para ***apt*** y se alarma frente a las alertas mostradas; se puede estar seguro de actualizar el sistema e instalar paquetes mediante ***apt.*** Por supuesto, siempre existe la posibilidad que algún atacante comprometa los servidores de Debian y se obtenga su clave privada.

7. HIDS, Host-based Intrusión Detection System

Un ***HIDS*** es, como su nombre indica, un sistema o serie de mecanismos por el cual se detectan cambios, accesos e irregularidades en una máquina. El beneficio que obtiene un administrador de sistemas utilizando este tipo de software es que será alertado e incluso puede que no se precise de su interacción para solventar un potencial problema detectado.

Anteriormente se introdujo ***Snort***, un ***NIDS*** capaz de analizar todo el tráfico de una red en busca de patrones sospechosos, accesos no deseados, etc. Ahora llega el tumo de ***OSSEC***, un ***HIDS*** muy potente y configurable licenciado bajo ***GPLv3***, compatible con multitud de sistemas.

OSSEC

OSSEC ofrece diversos mecanismos y opciones para proteger el sistema. Por lo tanto, antes de comenzar con la instalación del ***HIDS***, es necesario enumerar dichas características con la finalidad de aclarar ciertos conceptos, necesarios a la hora de configurar el entorno.

- ***Antiroot-kit.*** Esta funcionalidad se ofrece mediante un proceso intemo llamado ***rootcheck.*** La búsqueda se basará en una serie de patrones, firmas y ficheros definidos en unos ficheros de la configuración de ***OSSEC.***

- ***Análisis de integridad.*** Consiste en verificar que determinados archivos monitorizados no son modificados. Realmente no se monitorizan ficheros en concreto, sino directorios y los ficheros que estos contienen. En caso de no desear monitorizar determinados ficheros se pueden ignorar modificando la configuración. Para conseguir este objetivo se utilizan ***hashes*** del archivo y los permisos del mismo. En primer lugar se obtiene una lista de todos los ***hashes*** de los ficheros monitorizados y posteriormente de forma programada se realiza un análisis en busca de cambios de esos ficheros. En lugar de ejecutar un análisis de forma programada, es posible utilizar esta característica cuando se detecte la modificación de un fichero en tiempo real. Para esto último se usan las utilidades de ***inotify*** o ***inotify-tools.***

- Análisis de ***logs***, ***reglas*** y ***decoders.*** Es quizás una de las características más interesantes. ***OSSEC*** es capaz de analizar los ***logs***, obtener los nuevos mensajes de dichos ***logs*** y analizarlos para avisar sobre posibles problemas. Para el análisis de los ***logs*** se utilizan los llamados ***decoders***, que extraen la información necesaria de los ***logs*** como pueden ser direcciones IP, nombres de usuario, terminales del sistema, etc. Una vez que la información relevante ha sido extraída, esta es analizada de acuerdo a una serie de reglas que alertarán del suceso.

- ***Alertas.*** Son notificaciones que contienen la información de la regla que la han generado, así como otra información al respecto.

- ***Respuesta activa. OSSEC*** es muy potente y por defecto está configurado para detectar muchas situaciones de ataques, error, etc. A pesar de ello es posible que en algunas ocasiones se puedan generar falsos positivos. Por esa razón es peligroso utilizar la respuesta activa, que no es más que la ejecución de una serie de acciones cuando se lanza una alerta. Sólo debe utilizarse cuando se controle con total seguridad las alertas que ***OSSEC*** puede generar en un entorno.

Por otra parte, también a la hora de idear un entorno e instalar el ***HIDS***, es necesario saber que ***OSSEC*** dispone de diferentes modos de instalación y ejecución. No por ser un ***IDS*** de host quiere decir que no puedan gestionarse el estado y alertas de varias máquinas mediante un servidor. Para esta situación se utilizan los llamados agentes, cuya instalación en cada máquina "cliente" se verá a continuación.

Instalación y prueba de OSSEC

Para la instalación y prueba de ***OSSEC*** se supone un entorno simple con dos máquinas, una que actuará de servidor y otra de cliente en la que se instalará ***OSSEC*** como agente. Dicho entorno queda reflejado en la siguiente imagen.

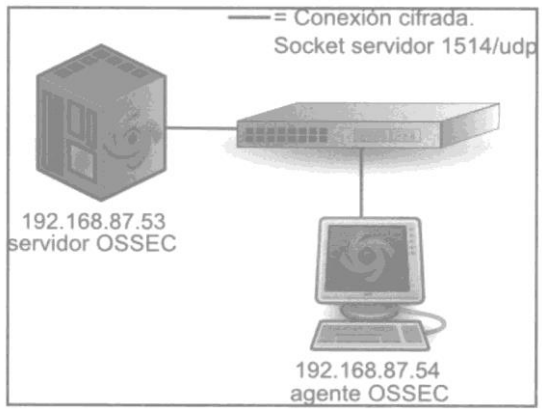


Fig 5.05: Entorno de pruebas para *OSSEC*.

A la hora de la instalación, es necesario tener en cuenta que este software no se ofrece desde los repositorios oficiales de Debian. Es por ello que habrá que descargar sus fuentes desde la web oficial del proyecto, http://www.ossec.net**.** La descarga se realizará en ambas máquinas, la que actuará de servidor ***OSSEC*** y el agente. De igual modo, como es necesario compilar las fuentes para su instalación, será necesario de disponer de las herramientas básicas de compilación. Adicionalmente, es muy recomendable instalar, siempre que el sistema lo soporte, las ***inotify-tools.*** Con esto último ***OSSEC*** será capaz de monitorizar la integridad de los ficheros en tiempo real. Para todo lo anterior, los pasos a seguir en cada una de las máquinas son los siguientes.

```
# apt-get install build-essential inotify-tools
```

Para la descarga de las fuentes habrá que descargar las fuentes de la última versión y los ***hashes*** para verificar que el paquete descargado es el correcto. Actualmente la última versión estable es la ***2.***7.

```
# wget http://www.ossec.net/files/ossec-hids-2.7.tar.gz
# wget http://www.ossec.net/files/ossec-hids-2.7-checksum.txt
# sha1sum -c ossec-hids-2.7-checksum.txt
```

```
ossec-hids-2.7.tar.gz: La suma coincide
# tar -xzf ossec-hids-2.7.tar.gz
```

Llega el momento de instalar el software en cada máquina según convenga. En ambos casos se omitirá el uso de ***Respuesta Activa*** por el hecho de empezar con un sistema sin configurar. Para la instalación es necesario situarse en el directorio recién extraído y ejecutar el ***script install.sh.***

Servidor

La instalación es muy sencilla, tan solo hay que responder a una serie de preguntas que afectan a la configuración y comportamiento de la aplicación. Es recomendable dejar el idioma de instalación en inglés, pues la traducción a veces deja bastante que desear.

```
# ./install.sh
...
1- What kind of installation do you want (server, agent, local, hybrid or help)?
server
  - Server installation chosen.

2- Setting up the installation environment.
 - Choose where to install the OSSEC HIDS [/var/ossec]:
    - Installation will be made at  /var/ossec .

3- Configuring the OSSEC HIDS.
  3.1- Do you want e-mail notification? (y/n) [y]:
   - What's your e-mail address? root@localhost
   - We found your SMTP server as: 127.0.0.1
   - Do you want to use it? (y/n) [y]:
   --- Using SMTP server:  127.0.0.1

  3.2- Do you want to run the integrity check daemon? (y/n) [y]:
   - Running syscheck (integrity check daemon).

  3.3- Do you want to run the rootkit detection engine? (y/n) [y]:
   - Running rootcheck (rootkit detection).

  3.4- Active response allows you to execute a specific
       ...
```

```
   - Do you want to enable active response? (y/n) [y]: n
     - Active response disabled.

  3.5- Do you want to enable remote syslog (port 514 udp)? (y/n) [y]:
   - Remote syslog enabled.

  3.6- Setting the configuration to analyze the following logs:
    -- /var/log/messages
    -- /var/log/auth.log
    -- /var/log/syslog
    -- /var/log/mail.info
    -- /var/log/dpkg.log
    -- /var/log/apache2/error.log (apache log)
    -- /var/log/apache2/access.log (apache log)
...
```

Después de especificar las características deseadas y pulsar <***Enter***> comenzará el proceso de instalación y compilación. Los directorios y ficheros relevantes para ***OSSEC*** son los siguientes.

- ***/etc/init.d/ossec.*** Es el script de inicio que controla la ejecución de ***OSSEC.*** Internamente llama al binario ***/var/ossec/bin/ossec-control.***
- ***/etc/ossec-init.conf.*** Este fichero es el encargado de establecer el directorio de trabajo de ***OSSEC***, en este caso ***/var/ossec*** y al mismo tiempo indica el tipo de ejecución de ***OSSEC*** con el parámetro ***TYPE.***
- ***/var/ossec/etc.*** Directorio con la configuración principal de ***OSSEC.*** El fichero principal es ***ossec.conf*** en el que se definen los tipos de acciones a realizar por ***OSSEC.*** Por acciones se entiende ***rootcheck***, análisis de integridad, directorio de reglas, etc. Dentro del directorio ***shared*** se encuentran las reglas para la detección de ***rootkits***, además de una serie de patrones para puntuar la configuración base del sistema. Esto último es el llamado test de CIS, ***Center for Internet Security.***
- ***/var/ossec/etc/decoder.xml.*** Es el fichero en el que se define el método para extraer información de los ***logs*** del sistema.
- ***/var/ossec/rules.*** Es el directorio en el que se almacenan las reglas para el procesado de cada ***log.*** Existen múltiples ficheros, cada uno de ellos con reglas que contienen descripciones y números de nivel de alerta. Un fichero de ejemplo podría ser***pamrules.xml,*** encargado de los sucesos de ***PAM*** visualizados en los ***logs.***
- ***/var/ossec/bin.*** Aquí es donde se encuentran todos los ejecutables necesarios para el funcionamiento de ***OSSEC.*** Es necesario tener en cuenta que este directorio no estará en el

PATH de ningún usuario, por lo que habrá que indicar esta ruta para ejecutar los comandos o estar situados dentro del directorio.

Se probará ***OSSEC*** con la configuración por defecto, que a veces incluso detecta eventos que no son necesarios. Es a modo de ver cómo funciona el sistema, aunque para un entorno en producción deben realizarse ajustes, crear reglas personalizadas, etcétera. Se inicia el agente como un servicio más.

```
# service ossec start
Starting OSSEC HIDS v2.7 (by Trend Micro Inc.)...
Started ossec-maild...
Started ossec-execd...
Started ossec-analysisd...
Started ossec-logcollector...
Started ossec-remoted...
Started ossec-syscheckd...
Started ossec-monitord...
Completed.
```

El servidor actúa como servidor y agente, por lo que también se verifican la integridad de ficheros, se buscarán ***rootkits***, etc.

Antes de pasar a la configuración del agente es necesario configurarlo y generar una ***PSK*** para poder transmitir la información entre el servidor y el agente de forma cifrada. Para todo ello se utiliza la aplicación ***/var/ossec/bin/manage agents*** con la opción ***A.***

```
squeetest:/var/ossec/bin# ./manage_agents
****************************************
* OSSEC HIDS v2.7 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: A
- Adding a new agent (use '\q' to return to the main menu).
  Please provide the following:
```

```
   * A name for the new agent: wheezy
   * The IP Address of the new agent: 192.168.87.54
   * An ID for the new agent[001]:
Agent information:
   ID:001
   Name:wheezy
   IP Address:192.168.87.54

Confirm adding it?(y/n): y
Agent added.
```

Ahora es necesario extraer la ***PSK*** asociada de forma automática al agente creado. Para ello se utiliza la opción ***E.***

```
****************************************
* OSSEC HIDS v2.7 Agent manager.     *
* The following options are available: *
****************************************
   (A)dd an agent (A).
   (E)xtract key for an agent (E).
   (L)ist already added agents (L).
   (R)emove an agent (R).
   (Q)uit.
Choose your action: A,E,L,R or Q: E

Available agents:
   ID: 001, Name: wheezy, IP: 192.168.87.54
Provide the ID of the agent to extract the key (or '\q' to quit): 001

Agent key information for '001' is:
AxIHdoZWV6eSAxOTIuMTY4Ljg3LjU0IDBkYTY1NDBlYzhiZDk1YWRhYzM4YmQzNzhhMj
BmM2NiNTZmZmFiYzMxYzM4YzRhNTViNGY0M2NhNDVlMWU2ZjU=
```

Con esa información puede pasar a configurarse el agente.

Agente

El proceso de instalación resulta idéntico salvo que habrá que indicar que se configura ***OSSEC*** en modo ***agente.***

```
# ./install.sh
...
1- What kind of installation do you want (server, agent, local, hybrid or help)?
agent
  - Agent(client) installation chosen.

2- Setting up the installation environment.
 - Choose where to install the OSSEC HIDS [/var/ossec]:
    - Installation will be made at  /var/ossec .

3- Configuring the OSSEC HIDS.
    3.1-  What's  the  IP  Address  or  hostname  of  the  OSSEC  HIDS  server?:
192.168.87.53
   - Adding Server IP 192.168.87.53

  3.2- Do you want to run the integrity check daemon? (y/n) [y]:
   - Running syscheck (integrity check daemon).

  3.3- Do you want to run the rootkit detection engine? (y/n) [y]:
   - Running rootcheck (rootkit detection).

  3.4 - Do you want to enable active response? (y/n) [y]: n
   - Active response disabled.

  3.5- Setting the configuration to analyze the following logs:
    -- /var/log/messages
    -- /var/log/auth.log
    -- /var/log/syslog
    -- /var/log/mail.info
    -- /var/log/dpkg.log
...
```

Una vez que finaliza la instalación es necesario agregar la ***PSK*** del agente generada anteriormente en el servidor. Para ello se ejecuta la aplicación ***manageagents***, pero esta vez en el cliente.

```
root@wizi:/var/ossec/bin# ./manage_agents
****************************************
* OSSEC HIDS v2.7 Agent manager.     *
* The following options are available: *
****************************************
   (I)mport key from the server (I).
   (Q)uit.
Choose your action: I or Q: I
...
Paste it here (or '\q' to quit):
MDAxIHdoZWV6eSAxOTIuMTY4Ljg3LjU0IDBkYTY1NDBlYzhiZDk1YWRhYzM4YmQzNzhhMjBmM2NiNT
ZmZmFiYzMxYzM4YzRhNTViNGY0M2NhNDVlMWU2ZjU=

Agent information:
   ID:001
   Name:wheezy
   IP Address:192.168.87.54
Confirm adding it?(y/n): y
Added.
```

El agente ya está configurado para comunicarse de forma segura con el servidor, por lo que únicamente queda iniciar el servicio ***ossec*** en esta máquina cliente.

```
root@wizi:~# service ossec start
Starting OSSEC HIDS v2.7 (by Trend Micro Inc.)...
Started ossec-execd...
Started ossec-agentd...
Started ossec-logcollector...
Started ossec-syscheckd...
Completed.
```

Para el caso del agente se edita la configuración para que los cambios en ficheros se notifiquen en tiempo real utilizando ***inotify.*** Para ello es necesario modificar la sección de configuración de <***syscheck***> dentro del fichero ***/var/ossec/etc/ossec.conf.*** En concreto se modificará la línea:

```
<directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
```

En este momento todos los ficheros de esos directorios se analizarán cada ***22*** horas, ***79200*** segundos establecidos por la línea.

```
<frequency>79200</frequency>
```

Para que las notificaciones sean en tiempo real la primera línea debe quedar del siguiente modo.

```
<directories realtime="yes" check_all="yes">/etc,/usr/bin,/usr/sbin</
directories>
```

Por lo tanto, los ficheros contenidos en los directorios ***/etc***, ***/usr/bin*** y ***/usr/sbin*** serán monitorizados en tiempo real. Para que los cambios surtan efectos es necesario reiniciar el servicio ***ossec.*** Por otra parte, es necesario destacar que esta funcionalidad no está disponible mientras ***syscheck*** está calculando los ***hashes*** iniciales de los archivos.

Ejemplos de funcionamiento

Inicialmente, tanto el servidor como el agente no son funcionales en su totalidad debido a la realización del inventariado de ficheros monitorizados con ***syscheckd.***

Las pruebas que se realizarán son las siguientes:

- Verificar la conexión con el agente utilizando la aplicación ***agent control.***
- Visualizando el log ***/var/ossec/logs/alerts/alerts.log:***
  - ***Password*** incorrecto utilizando ***sudo*** en la máquina cliente, ***wheezy.***
  - Modificación del fichero ***/etc/passwd.***

Los parámetros ***-le*** de ***agent control*** muestran los agentes activos.

```
squeetest:/var/ossec/bin# ./agent_control -lc
OSSEC HIDS agent_control. List of available agents:
   ID: 000, Name: squeetest (server), IP: 127.0.0.1, Active/Local
   ID: 001, Name: wheezy, IP: 192.168.87.54, Active
```

Atendiendo al log, con las pruebas indicadas se obtienen los siguientes resultados.

```
** Alert 1361730388.11632: - pam,syslog,authentication_failed,
2013 Feb 24 19:26:28 (wheezy) 192.168.87.54->/var/log/auth.log
Rule: 5503 (level 5) -> 'User login failed.'
```

```
Feb 24 19:26:27 wizi sudo: pam_unix(sudo:auth): authentication failure;
logname=carlos uid=1001 euid=0 tty=/dev/pts/0 ruser=carlos rhost= user=carlos
...
** Alert 1361731714.18358: mail - ossec,syscheck,
2013 Feb 24 19:48:34 (wheezy) 192.168.87.54->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/passwd'
Size changed from '1748' to '1749'
Old md5sum was: '22805e608424d1b3c2ea79a0dfacaa15'
New md5sum is : 'eae77e12463df8100d9cd17ed9b4c537'
Old sha1sum was: '3bc5baf7cee1a3334f5a61d76821cba6440e6dc9'
New sha1sum is : '7282a866f58208f8c90112b647c73bdc17c54015'
```

La primera de las alertas no será notificada por e-mail ya que el nivel de alerta que tiene es 5, y ***OSSEC*** está configurado por defecto para notificar vía e-mail las alertas que superen el nivel 6. Este comportamiento es configurable en el fichero ***/var/ossec/etc/ossec.conf*** del servidor, con las siguientes líneas.

```
<alerts>
    <log_alert_level>1</log_alert_level>
    <email_alert_level>7</email_alert_level>
</alerts>
```

En el caso de la modificación del fichero /***etc/passwd***, el correo que se recibe sería el siguiente.

```
From ossecm@squeetest Sun Feb 24 19:47:06 2013
Envelope-to: root@localhost
Delivery-date: Sun, 24 Feb 2013 19:47:06 +0100
To: <root@localhost>
From: OSSEC HIDS <ossecm@squeetest>
Date: Sun, 24 Feb 2013 19:47:05 +0100
Subject: OSSEC Notification - (wheezy) 192.168.87.54 - Alert level 7

OSSEC HIDS Notification.
2013 Feb 24 19:46:52

Received From: (wheezy) 192.168.87.54->syscheck
```

```
Rule: 550 fired (level 7) -> "Integrity checksum changed."
Portion of the log(s):

Integrity checksum changed for: '/etc/shadow'
Size changed from '1307' to '1308'
Old md5sum was: '56827c77f5e8a83694b6d714b4c31257'
New md5sum is : '08fdfe82a91a2f72c393316df8db8777'
Old sha1sum was: '171e92815c037298d17c714b80899b4926b6d0b0'
New sha1sum is : '9303505960fc8325f54d2e80e9f0d90c70a140c5'
```

Se puede comprobar que aun habiendo utilizado la configuración por defecto casi por completo, resulta muy interesante y efectiva la funcionalidad que proporciona este software. Adicionalmente puede utilizarse en conjunción a alguna interfaz web, para que los resultados sean más visuales y sencillos de interpretar. Un ejemplo de esto último es utilizando la interfaz ***OSWUI***, proporcionada desde la propia web oficial de ***OSSEC.***

Capítulo VI
Fortificación de un entorno LAMP

LAMP son las siglas con las que se identifica a una infraestructura de servidor para alojar aplicaciones web. El nombre proviene de la unión ***Linux***, ***Apache***, ***MySQL*** y ***PHP***\ componentes correspondientes al sistema operativo, servidor web, servidor de bases de datos e intérprete de Scripts en ***PHP.*** A pesar de que este entorno solo sirva aplicaciones ***PHP***, la elección no ha sido al azar sino por la popularidad del mismo. Las estadísticas demuestran que a día de hoy la mayoría de servidores que alojan aplicaciones web funcionan utilizando los componentes mencionados. En este capítulo se proporcionarán algunos de los muchos parámetros, mecanismos y configuraciones existentes para fortificar este tipo entornos ante ataques de tipo ***SQLi, XSS, RFI***, ***LFI, path traversal,*** etc.

1. Instalación de un entorno LAMP

La instalación en un entorno Debian resulta extremadamente sencilla desde los repositorios oficiales. Para ello como de costumbre se utilizará ***api.*** Con la siguiente línea se obtendría un entorno ***LAMP*** completamente operativo desde el inicio.

```
# apt-get install apache2 libapache2-mod-php5 mysql-server
```

Durante la instalación de todos los paquetes se preguntará por el password de administrador de MySQL. Deberá introducirse un password robusto para proteger el acceso al gestor de bases de datos.

La orden anterior habrá instalado Apache2 en modo prefork, el módulo de PHP en su versión 5 para Apache y el servidor MySQL. Por supuesto, todo ello acompañado de algunas dependencias que apt instalará de forma automática. A título informativo, el modo prefork de Apache es un modo de funcionamiento en el que Apache irá creando procesos hijos por cada petición. Resulta menos eficiente pero se instala por defecto por compatibilidad con PHP. A pesar de ello es posible disponer de PHP y Apache2 en su modo worker para mejorar el rendimiento. En cualquier caso no se discutirá esa opción.

Para los tres componentes instalados los principales ficheros de configuración son los siguientes.

Servicio	Fichero	Descripción
MySQL	/etc/mysql/my.cnf	Es el fichero de configuración clásico y de opciones globales para *MySQL*. Contiene tanto la configuración del servicio *MySQL* como utilizada por clientes y otras utilidades para el mantenimiento y gestión de la base de datos.
	/etc/mysql/conf.d/	Habitual en las distribuciones basadas en Debian, es el directorio donde podrán colocarse configuraciones fragmentadas. Todos los ficheros terminados en *.cnf* serán interpretados por *MySQL* para su configuración.
	/etc/mysql/debian.cnf	Se trata de un fichero propio de distribuciones Debian. En él se definen parámetros para el mantenimiento no interactivo de las bases de datos *MySQL*. Por ejemplo para comprobar corrupción en bases de datos. Este fichero debe disponer de permisos 0600, con *root* como propietario y grupo, ya que de lo contrario cualquier usuario podría obtener credenciales con privilegios administrativos en *MySQL*.
PHP	/etc/php5/apache2/php.ini	Es el fichero donde se definen todos los parámetros para modificar el comportamiento y la funcionalidad de *PHP*. Está muy documentado y por lo tanto posee mucho contenido. Es recomendable crear uno limpio eliminando comentarios a modo de documentación.
	/etc/php5/conf.d/	Es el directorio del que se leerá la configuración para *PHP* de forma fragmentada. Con la instalación realizada existen un par de ficheros de configuración, siendo el más relevante el llamado *suhosin.ini* que será mencionado más adelante.
Apache	/etc/apache2/setenv	Aquí es donde se establecerán las variables principales de funcionamiento de *Apache*. El *front-end apache2ctl*, utilizado para el control del demonio *apache2*, será el que utilice esta configuración. Entre otros parámetros, se pueden encontrar desde el usuario que ejecutará *Apache*, hasta algunos directorios de trabajo.
	/etc/apache2/apache2.conf	Es el fichero de configuración principal de *Apache* en distribuciones Debian. En él se definen directivas básicas tanto de log, configuración de conexiones máximas, etc.; pero además se hacen diversos *Include* a otros ficheros de configuración.

Servicio	Fichero	Descripción
Apache	/etc/apache2/mods-enabled/	En este directorio se encuentran las referencias a los módulos habilitados para *Apache*. Existen ficheros terminados en *.load* y en ocasiones una pareja en *.conf*. El primer tipo de archivos incluye una sentencia con la directiva *LoadModule* que cargará el módulo de *Apache* especificado. Como es de imaginar, los ficheros *.conf* se utilizan para modificar ciertos parámetros en la configuración. En este directorio se crearán enlaces simbólicos de los ficheros *.load* y *.conf* contenidos en el directorio */etc/apache2/mod-available/*. Se puede realizar manualmente, sin embargo existen las utilidades *a2enmod* y *a2dismod* para habilitar y deshabilitar módulos respectivamente.
	/etc/apache2/sites-enabled/	Es el directorio análogo al anterior pero destinado a los diferentes sitios alojados por *Apache*. Permite segmentar las configuraciones por sitios. Cada fichero contendrá las típicas directivas de configuración de *Apache* como *VirtualHost*, *Directory*, *Order*, *Allow*, etc. Al igual que en el caso de los módulos los ficheros son enlaces simbólicos hacia los ficheros contenidos en el directorio */etc/apache2/sites-available/*. Para habilitar o deshabilitar configuraciones de sitios es posible utilizar las aplicaciones *a2ensite* y *a2dissite* respectivamente.
	/etc/apache2/httpd	Se trata del fichero de configuración tradicional de *Apache*. En el caso de Debian puede seguir utilizándose aunque está vacío por defecto.
	/etc/apache2/ports.conf	En este fichero se definen los puertos de escucha de *Apache* de forma global. Por defecto se establece el puerto 80 y 443 para conexiones *HTTP* y *HTTPS*.
	/etc/apache2/conf.d/	Se dispone de otro directorio donde establecer otro tipo de configuraciones. Existen varios ficheros por defecto pero el más interesante es el llamado *security*.

Adicionalmente a los ficheros anteriores, en el caso de ***MySQL*** la mayoría de la configuración está contenida en las propias bases de datos ***mysql*** e ***informations chema.***

De igual modo, en el caso de ***PHP*** sus opciones y parámetros pueden ser configurados desde los ficheros de configuración de ***Apache***, ya que este carga ***PHP*** como un módulo.

A continuación se enumerarán algunos de los parámetros más comunes para tratar de fortificar cada uno de los componentes utilizados. Adicionalmente es muy recomendable visitar las documentaciones oficiales en línea de cada uno de ellos.

- ***MySQL*** - ***http://dev.mysql.com/doc/index.html***
- ***PHP*** - ***http://www.php.net/manual/es/***
- ***Apache*** - ***http://httpd.apache.org/docs/***

2. MySQL

Una vez instalado el gestor de bases de datos ***MySQL*** en Debian cabe decir que por defecto la instalación es bastante segura, sin embargo a continuación se verán algunas opciones modificables para aumentar la seguridad de dicha configuración.

Existen configuraciones aplicables mediante ficheros y en cambio otras directamente en la base de datos utilizando el cliente ***mysql*** o bien el comando ***mysqladmin.*** Las opciones modificadas en la propia consola de ***MySQL*** estarán precedidas por ***mysql>***, que es el ***prompt*** por defecto.

Dirección de escucha

Se trata de una directiva que establecida a un valor seguro en la instalación desde Debian. Por defecto ***MySQL*** escuchará en local, por lo que no aceptará conexiones que provengan de otras máquinas independientemente de la configuración de un cortafuegos. El parámetro en cuestión es el siguiente:

```
bind-address=127.0.0.1
```

Sólo se escuchará en la interfaz con dirección IP ***127.0.0.1***, que es ***localhost.*** Desde una máquina externa aparecería el siguiente error al intentar establecer la conexión con esta máquina, ***192.168.87.55.***

```
# mysql -uroot -pP4ssW0rd -h192.168.87.55
ERROR 2003 (HY000): Carr*t connect to MySQL server on ⁽192.168.87.55⁾ (111)
```

Carga de ficheros locales

En las aplicaciones web se pueden explotar diversos tipos de ataque, como los mencionados en la introducción de este capítulo. Si por ejemplo un atacante descubriese la forma de explotar un ***SQLi*** y quisiera acceder a ficheros del sistema, podría obtener información de los archivos accesibles por el usuario que ejecute el demonio ***MySQL.***

Por defecto, desde la consola de ***MySQL*** podría realizarse lo siguiente.

```
mysql> select load_file('/etc/passwd');
...
carlos:x:1000:1000:Carlos,,,:/home/carlos:/bin/bash
mysql:x:103:106:MySQL Server,,,:/var/lib/mysql:/bin/false
...
```

Además de ***load Jile*** existe el método ***LOAD DATA LOCAL INFILE.***

Para deshabilitar la carga de ficheros desde ***MySQL*** y evitar una posible fuga de información desde el sistema de ficheros mediante un ***SQLi,*** es necesario establecer la siguiente configuración en el fichero ***/etc/mysql/my.cnf***

```
...
[mysqld]
...
local-infile    = 0
secure-file-priv        = /dev/null
...
```

También existe la posibilidad de eliminar esta funcionalidad para usuarios en lugar de hacerlo globalmente. Para ello hay que revocar el privilegio ***file_priv*** a los usuarios deseados. Es posible verificar qué usuarios disponen del privilegio con la siguiente consulta.

```
mysql> select user,host,file_priv from mysql.user;
```

Renombrar el usuario root

Realmente no es una medida de seguridad como tal, pero hará más difícil a un atacante la labor de encontrar al usuario administrador de la base de datos. Para renombrar el usuario ***root*** como ***Carlos*** se podría hacer del siguiente modo.

```
mysql> update mysql.user set user="carlos" where user="root";
mysql> flush privileges;
```

Comprobar existencia de usuarios anónimos

En numerosas instalaciones de ***MySQL*** se crean usuarios anónimos por defecto que podrían ser utilizados para extraer información. Para eliminar esta problemática se puede comprobar la existencia de usuarios anónimos del siguiente modo.

```
mysql> select user usuarios,host from mysql.user where user="";
```

En el caso de obtener algún resultado deberían eliminarse dichos usuarios o bien limitar sus privilegios al mínimo.

Controlar los privilegios de los usuarios

Una de las medidas fundamentales para proteger la privacidad de los datos frente a ataques consiste en disponer de diferentes usuarios para cada cometido. Por su parte, cada usuario deberá tener determinados privilegios dependiendo del nivel de funcionalidad del que se le quiera dotar.

Con las siguientes acciones, se obtendrán los diferentes usuarios y sus privilegios.

```
mysql> select distinct(grantee) from information_schema.user_privileges;
```

El resultado de esa orden mostrará líneas como ***'root '@ 'localhost*** \ o lo que es lo mismo los usuarios junto al host. Para comprobar los privilegios de ese usuario en concreto podría realizarse la siguiente consulta.

```
mysql> SHOW GRANTS FOR 'root'@'localhost';
```

La asignación de los privilegios dependerá de la funcionalidad deseada en cada caso.

mysql_secure_installation

Se trata de un ***script*** encargado de asegurar determinados aspectos de ***MySQL.*** En concreto buscará usuarios anónimos, establecerá el ***password*** del usuario administrador, eliminará las bases de datos ***test***, etc. Es recomendable ejecutar este ***script***, ya sea antes o después de las configuraciones ya mencionadas.

Otras configuraciones recomendadas son la utilización de una jaula ***chroot*** y la utilización de un usuario para la ejecución de ***MySQL*** sin privilegios. En Debian por defecto se utiliza el usuario ***mysql.*** Por otra parte, para finalizar con ***MySQL***, resulta evidente que los permisos de los ficheros de las bases de datos, así como los binarios pertenecientes a ***MySQL***, deben ser exclusivos para el usuario ***mysql.***

3. PHP

En numerosas ocasiones un sistema debe publicar aplicaciones ***PHP*** realizadas por terceros que no siempre están programadas correctamente. Son muchos los Scripts en que dejan puertas abiertas frente a ataques comunes. Para tratar de controlar determinados aspectos de ***PHP*** existen muchos parámetros para modificar el comportamiento de este intérprete. Algunos de los más utilizados son los siguientes.

expose_php

Cuando un atacante está inspeccionando un sistema o recurso remoto en busca de vulnerabilidades, es casi obligatorio realizar ***fingerprinting.*** En el caso de un servidor web, cuando se realiza una captura con ***Wireshark*** en el caso de este entorno ***LAMP*** se obtiene la siguiente respuesta ***HTTP.***

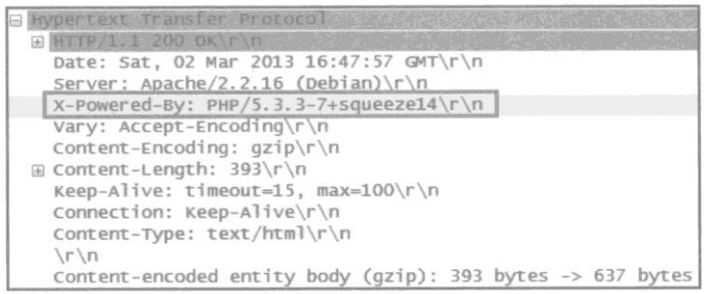


Fig 6.01: Respuesta *HTTP* con *expose_php* activado.

Como se puede observar se obtiene la versión de ***PHP.*** Para evitar que esa información sea pública es posible evitar que se envíe esa cabecera, mediante el siguiente parámetro en el fichero ***/etc/php/apache2/php. ini.***

```
expose_php = Off
```

displayerrors

En la fase de desarrollo, cuando se está creando la aplicación y depurando su funcionamiento resulta vital obtener mensajes de errores y alertas; sin embargo en un entorno en producción no debería mostrarse esa información. Son muchos los sitios web que revelan información sensible cuando se fuerzan errores manipulando parámetros de la aplicación de forma inesperada.

Para deshabilitar la salida de errores el parámetro ***display errors*** debe estar como a continuación. Es el estado por defecto en Debian.

```
display_errors = Off
```

openbasedir

Uno de los ataques más comunes en aplicaciones web son los llamados ***Path Traversal.*** El ejemplo más clásico para esto consiste en tratar de conseguir ficheros con información sensible como / ***etc/passwd.*** Es posible configurar ***PHP*** para que sus operaciones con ficheros estén contenidas en una ubicación determinada.

```
open_basedir = /var/www
```

Para comprobar su funcionamiento se ejecutará el siguiente ***script PHP*** ubicado en el fichero ***/vari www/index.php.***

```
<?php include $_GET['recurso'] ?>
```

Simulando el acceso a ***/etc/passwd*** mediante http://l ***92.168.87.55/?recurso =/etc/passwd,*** se mostraría la siguiente información en el loe de error de ***Avache***. ***error loe.***

```
[Sat Mar 02 19:08:51 2013] [error] [client 192.168.87.3] PHP Warning:  include():
open_basedir restriction in effect. File(/etc/passwd) is not within the allowed
path(s): (/var/www) in /var/www/index.php on line 1
```

disablefunctions

Con la finalidad de proteger la máquina donde se ejecutan las aplicaciones web, es posible deshabilitar determinadas funciones de ***PHP.*** Esto puede resultar útil en situaciones en las que un atacante consigue inyectar algún tipo de ***shell*** en el contexto de una aplicación web. Denegando el acceso a determinadas funciones se podrá limitar el impacto del ataque.

```
disable_functions = phpinfo, system, exec, shell_exec, ini_set, dl, eval
```

En caso de que un script tratase de utilizar alguna de las funciones anteriores, aparecería el siguiente mensaje en el error.log de ***Apache.*** Por ejemplo para ***ini_set.***

```
[Sat Mar 02 19:51:11 2013] [error] [client 192.168.87.3] PHP Warning:  ini_set()
has been disabled for security reasons in /var/www/index.php on line 3
```

Es importante evitar que se utilice la función ***inijset*** en los ***Scripts*** de ***PHP*** pues se estará permitiendo sobrescribir los valores establecidos en el fichero ***/etc/php/apache2/php.ini.***

Deshabilitar RFI

En ***PHP*** existe la posibilidad de incluir recursos externos a la propia máquina donde se está ejecutando la aplicación web. En numerosas ocasiones existen ***scripts*** mal programados que mediante parámetros ***GET*** o ***POST*** permitirían la inclusión de código malicioso por parte de un atacante. Existen dos opciones para deshabilitar este comportamiento, tanto para incluir código de un ***script***, como para abrir ficheros.

```
allow_url_fopen = Off
allow_url_include = Off
```

Por defecto la segunda opción está en ***Off.***

Otras opciones del fichero ***php.ini*** permitirán limitar el uso de recursos por parte de ***PHP***, evitar la subida de ficheros, forzar que ***JavaScript*** no pueda acceder a las ***cookies*** con ***session.cookie_ httponly***, etc. Todos ellos permitirán mejorar la configuración del entorno atendiendo a la seguridad del mismo.

Suhosin

Es un componente diseñado para la protección de instalaciones ***PHP.*** La web del proyecto es http://www.hardened-php.net/suhosin y en ella se encontrarán detallados todos los parámetros de configuración. En Debian el paquete es ***php5-suhosin*** y se instala por defecto como dependencia de los componentes ***LAMP.***

Sin necesidad de modificar ningún parámetro de configuración, este módulo empezará a funcionar evitando determinados ataques o usos no debidos de ***PHP,*** cifrado de sesiones, filtrado de caracteres de entrada, límites de parámetros, etc.

Un ejemplo de ***Suhosin*** funcionando se aprecia cuando ha existido un intento de ***RFI***, para ello se atiende al log ***/var/log/syslog***, que es donde se almacenan los mensajes generados por ***Suhosin*** de forma predeterminada.

```
Mar 2 15:03:28 test suhosin[10179]: ALERT - Include filename ('http://192.168.87.53/
recurso') is an URL that is not allowed (attacker '192.168.87.3', file '/var/www/
index.php', line 2)
```

Se muestra que la página de carga no está permitida ya que no está incluida en una ***whitelist*** de ***Suhosin***, vacía por defecto.

El fichero de configuración se encuentra por defecto en ***/etc/php/conf.d/suhosin.*** Un ejemplo de parámetro a modificar sería el siguiente.

```
suhosin.executor.include.max_traversal = 2
```

Indica cuántos saltos hacia arriba se permitirán como máximo a la hora de abrir ficheros. Evitaría ataques del tipo ***http://l****92.1****68.8****7.55****//recurso=../../etc/passwd,*** mostrando en el *syslog* la siguiente información.

```
Mar  2 14:33:15 test suhosin[7606]: ALERT - Include filename ('../../etc/passwd')
contains too many '../' (attacker '192.168.87.3', file '/var/www/index.php', line
3)
```

En caso de no necesitar el funcionamiento activo de ***Suhosin*** es posible deshabilitarlo con la siguiente línea de configuración. Aunque esto no es recomendable a menos que se muestren incompatibilidades con las aplicaciones web.

```
suhosin.simulation = on
```

Suhosin seguirá mostrando información en los ***logs*** del sistema pero sin embargo no actuará para detener los ataques o usos no permitidos de ***PHP*** que detecte.

```
Mar  2 15:35:08 test suhosin[10371]: ALERT-SIMULATION - Include filename
('http://192.168.87.53/recurso') is an URL that is not allowed (attacker
'192.168.87.3', file '/var/www/index.php', line 10)
```

Todas las opciones y mecanismos mostrados anteriormente proporcionarán cierta seguridad a la hora de ejecutar aplicaciones ***PHP*** en una máquina. Sin embargo, es recomendable utilizar métodos seguros cuando se programa una aplicación. Para este tipo de situaciones existe ***PHPIDS*** por ejemplo. Consiste en un conjunto de recursos para hacer más segura la programación en ***PHP***, descartando ataques mediante patrones, filtros, etc.

4. Apache

En estos instantes el servidor web ***Apache*** estará instalado y listo para servir cualquier contenido estático o dinámico utilizando el intérprete ***PHP.*** Sin embargo es posible mejorar la configuración por defecto para obtener un servidor web más seguro y dificultar en cierta medida a los atacantes, cuyos objetivos suelen ser entornos de este tipo.

Configuraciones globales

Las siguientes configuraciones afectarán al comportamiento global de ***Apache*** y se aplicará en los diferentes ficheros de configuración mencionados anteriormente. Por otra parte, se hará uso de la herramienta ***a2dismod*** para deshabilitar ciertos módulos.

Usuario y grupo de ejecución

Un aspecto que se debe tener en cuenta en la ejecución de servicios pasa por controlar qué usuario los ejecuta. La idea consiste en que un servicio sea ejecutado por un usuario dedicado a él, con los mínimos privilegios posibles, de modo que ante un fallo en la aplicación el impacto sea el menor posible.

En el caso ***de Apache*** en Debian, el usuario y grupo que ejecuta el servicio es ***www-data.*** De cualquier modo, siempre es recomendable verificar esta configuración observando los siguientes parámetros del fichero de configuración ***/etc/apache2/apache2.conf.***

```
User www-data
Group www-data
```

Deshabilitar módulos innecesarios

Apache puede estar compilado de forma estática, con todas las funcionalidades incluidas, o bien con el soporte para los llamados módulos ***DSO*** que es la opción por defecto en este caso. Es posible que

algunos de los módulos que están configurados para iniciarse por defecto no sean necesarios, por lo que se recomienda desactivarlos. En este caso se hará uso de la herramienta ***a2dismod.***

```
root@test:~# a2dismod
Your choices are: alias auth_basic authn_file authz_default authz_groupfile authz_
host authz_user autoindex cgi deflate dir env mime negotiation php5 reqtimeout
setenvif status
Which module(s) do you want to disable (wildcards ok)?
autoindex cgi
Module autoindex disabled.
Module cgi disabled.
Run '/etc/init.d/apache2 restart' to activate new configuration!
```

Para este ejemplo se han deshabilitado los módulos ***autoindex*** y ***cgi.*** El primero de ellos permite listar el contenido de los directorios servidos por ***Apache*** cuando no existe un fichero ***índex.*** El segundo permite el uso de CGI, que se desactiva en este caso por no utilizarlo. Como indica ***a2dismod***, una vez que se reinicie el servidor web los cambios serán efectivos.

Deshabilitar información ofrecida por el servidor

Cuando se realiza un petición al servidor web ***Apache***, en las cabeceras que devuelve como parte de la respuesta se mostrará información sobre la versión de ***Apache***, versión de ***PHP,*** etc. De igual modo sucede cuando se fuerza un error de página inexistente, que dependiendo de la configuración del servidor se mostrará la versión del software y las principales características instaladas.

Para evitar esta situación se modificarán un par de parámetros en la configuración. Realmente no ofrecen seguridad como tal, pero sí dificulta a un potencial atacante la identificación del entorno al que se enfrenta.

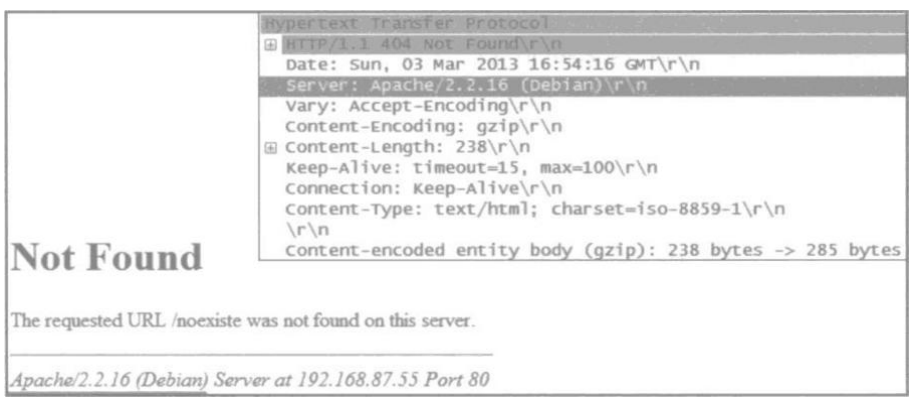


Fig 6.02: Error en el navegador y cabecera *HTTP* desde *Wireshark.*

En el fichero ***/etc/apache2/confd/security*** existen dos directivas llamadas ***ServerTokens*** y ***ServerSignature*** con valores ***OS*** y ***On*** respectivamente. La idea es que queden del siguiente modo.

```
ServerTokens ProductOnly
ServerSignature Off
```

Fig 6.03: Error en el navegador y cabecera *HTTP* después de las modificaciones.

Como se aprecia en ambas imágenes el cambio es significativo. A partir de este momento no se distinguirá la versión de ***Apache*** ejecutada.

Configuraciones por contexto

Para mostrar algunas de las configuraciones aplicables a cada contexto o sitio en ***Apache*** se utilizará el sitio habilitado por defecto. El contenido de la configuración por defecto se encuentra en el fichero ***/etc/apache2/sites-available/default***, que se encuentra habilitado al existir un enlace simbólico hacia él en el directorio ***/etc/apache2/sites-enabled.***

El contenido del fichero es el siguiente.

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
```

```
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Existen una gran variedad de parámetros para modificar el comportamiento de los sitios servidos por ***Apache.*** Algunos de los más importantes, contenidos en el fichero, son los siguientes.

Options

Con la directiva ***Options*** es posible especificar determinadas funcionalidades del servidor web Apache dentro del contexto ***Directory*** en el que se defina. Algunas de las opciones son.

- ***FollowSymLinks.*** Hará que ***Apache*** siga los enlaces simbólicos que se encuentren dentro del directorio. En el ejemplo de configuración mostrado, cualquier enlace simbólico que se encuentre dentro de ***/var/www/*** o niveles inferiores será leído. Se debe tener especial cautela con esta orden pues podría proporcionar acceso a información sensible. Un ejemplo grave podría ser un enlace como ***In -s /etc/ /var/www/etc.***

- ***SymLinksIfOwnerMatch.*** A diferencia de la configuración anterior solo se seguirán enlaces simbólicos que tengan el mismo propietario que el recurso enlazado. Si los propietarios de todos los ficheros y enlaces dentro de ***/var/www*** se estableciesen siempre como ***www-data*** no habría excesivo peligro.

- ***MultiViews.*** Tratará de ofrecer un recurso indicado si este no dispone de una extensión. Se entregará el recurso que exista probando con las extensiones definidas en ***/etc/mime. types.*** Si por ejemplo existiera un fichero debian.png en la raíz, se podrá escribir la dirección http://servidor/debían para cargar la imagen.
- ***Indexes.*** Mostrará el contenido del directorio si no existe un fichero ***Índex.*** En este caso no surtirá ningún efecto pues previamente se desactivó el módulo ***autoindex.***
- ***None.*** Es la opción que debería utilizarse siempre que no se necesite ninguna otra opción.

Adicionalmente cada una de las opciones puede estar precedida por los símbolos + o - fusionar las opciones con otras ya definidas o heredadas.

ACL

Es posible restringir el acceso al contenido para determinadas direcciones o redes. Esto puede resultar útil cuando el mismo servidor que ofrece contenido público se utiliza para mostrar contenido que solo debería ser accesible por una intranet por ejemplo.
Para conseguir esta funcionalidad se utilizan las directivas ***Order, Allow, Deny.*** Para explicar el funcionamiento se atiende a la configuración anterior.

```
Order Allow,Deny
Allow from all
```

Según el orden establecido en ***Order*** se evaluará antes la entrada ***Allow*** o ***Deny.*** En ese caso se evalúa en primer lugar la directiva ***Allow***, que al estar definida de ese modo permitirá el acceso a todos. Otro ejemplo sería el siguiente.

```
Order Allow,Deny
Allow from 192.168.87.0/24 informatica64.com
Deny from all
```

Se permitirá el acceso a la subred ***192.168.87.0/24*** y a la dirección del dominio ***informatica64.com.*** En caso de necesitar denegar en primer lugar bastaría con modificar la directiva ***Order.***

```
Order Deny,Allow
Deny from 10.0.1.23
Allow from all
```

Con esa última configuración se denegará el acceso a la dirección IP ***10.0.1.23.***

htaccess

Los ficheros ***.htaccess*** se colocan dentro de los directorios donde se desee aplicar una configuración específica definida por el propio usuario o ***webmaster*** del portal y no por el administrador de ***Apache.*** Sin duda es una funcionalidad que ofrece gran flexibilidad en cuanto a configuración se refiere, pero aumenta el riesgo de introducción de errores en la configuración de ***Apache.***

Por defecto la opción que determina el uso de los ficheros ***.htaccess*** es ***AllowOverride***, que está desactivada por defecto. Debe permanecer en ese estado a menos que sea necesario activarla. En caso de habilitar los ficheros ***.htaccess***, es posible definir opciones de configuración de Apache que no sean sobreescritas por las configuraciones realizadas en estos ficheros mediante categorías disponibles para ***AllowOverride.*** Toda esta información puede ser consultada en la documentación oficial de ***Apache.***

El fichero de sitio web proporcionado por defecto podría quedar del siguiente modo aplicando las directivas mencionadas.

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options None
                AllowOverride None
                Order deny,allow
                deny from all
        </Directory>

        <Directory /var/www/>
                Options None
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/default/error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/default/access.log combined
</VirtualHost>
```

Un aspecto destacable reside en que este contexto por defecto dispondrá de sus propios ficheros de log, con lo que será más sencillo detectar accesos.

mod_security

Se trata de un módulo que permite filtrar peticiones maliciosas recibidas por el servidor web ***Apache.*** Se considera como un ***IDS*** para aplicaciones web y es posible registrar los ataques recibidos en el sistema, así como prevenirlos. Mediante los filtros y patrones que implementa podrá detectar ataques ***SQLi*** y ***XSS*** entre muchos otros, siendo una eficaz protección ante aplicaciones web inseguras.

Se diferencian dos partes en ***modsecurity.*** La primera de ellas es el módulo en sí y la segunda son las reglas que contienen los filtros. Las reglas reciben el nombre de ***CRS***, ***Core Rule Set.***

Instalar mod_security

Es posible instalar ***mod security*** desde los repositorios oficiales de Debían, pero las versiones están desactualizadas al igual que ocurría con ***Snort.*** Es por tanto recomendable descargar las fuentes del módulo desde la página oficial del proyecto, compilarlo e instalarlo.

En primer lugar será necesario preparar el sistema con una serie de dependencias tal y como se especifica en la documentación oficial.

```
# apt-get install build-essential autoconf automake libapr1-dev apache2-threaded-
dev libpcre3-dev libxml2-dev liblua5.1-0-dev libcurl4-openssl-dev
```

Una vez instalados los paquetes llega el momento de descargar las fuentes de ***mod security.*** Para ello se realizan las siguientes acciones.

```
# wget http://www.modsecurity.org/tarball/2.7.2/modsecurity-apache_2.7.2.tar.gz
#  wget  http://www.modsecurity.org/tarball/2.7.2/modsecurity-apache_2.7.2.tar.
gz.md5
# md5sum -c modsecurity-apache_2.7.2.tar.gz.md5
modsecurity-apache_2.7.2.tar.gz: La suma coincide
# tar xzf modsecurity-apache_2.7.2.tar.gz
```

Una vez descargado y extraído, el paquete se compila e instala del siguiente modo.

```
# cd modsecurity-apache_2.7.2/
# ./autogen.sh
# ./configure
# make
# make install
```

```
# cp /usr/local/modsecurity/lib/mod_security2.so /usr/lib/apache2/modules/
```

En este instante el módulo está compilado y listo para que ***Apache*** pueda utilizarlo. Tan solo es necesario descargar las reglas y crear los ficheros de configuración para ***Apache.***

Preparación de Apache

Antes de comenzar con la configuración de ***Apache*** es necesario descargar reglas actualizadas desde la web de ***modsecurity.*** Tanto para la descarga, como para el almacenamiento de las reglas se creará un directorio dentro de la configuración de ***Apache.*** Por lo tanto habría que realizar las siguientes acciones.

```
# mkdir /etc/apache2/modsec
# cd /etc/apache2/modsec
# wget https://github.com/SpiderLabs/owasp-modsecurity-crs/tarball/master \
-O crs-`date +%Y%m%d`.tgz
# tar xzf crs-*.tgz
# cd SpiderLabs*
# cp -r base_rules/ ../
# cp modsecurity_crs_10_setup.conf.example ../ modsecurity_crs_10_setup.conf
# cd ..
# ls -l
base_rules
crs-20130303.tgz
modsecurity_crs_10_setup.conf
SpiderLabs-owasp-modsecurity-crs-52370fc
```

Ahora que las reglas están en el directorio ***/etc/apache2/modsec***, solo faltan crear los ficheros ***.load*** y ***.conf*** dentro de los módulos disponibles de ***Apache.*** De este modo ***Apache*** será capaz de cargar el módulo y su configuración. Para facilitar la tarea de la configuración existe una configuración base muy útil para comenzar con ***mod security.*** Se encuentra dentro del directorio de las fuentes extraídas previamente.

```
# cp modsecurity.conf-recommended /etc/apache2/mods-available/modsecurity.conf
```

Una vez copiado el fichero habrá que editarlo agregando las rutas donde ***mod security*** podrá encontrar las reglas previamente extraídas.

```
Include /etc/apache2/modsec/*.conf
Include /etc/apache2/modsec/base_rules/*.conf
```

Para finalizar con la configuración de ***Apache*** tan solo resta la creación del fichero ***/etc/apache2/mods-available/modsecurity.load***, que deberá contener las siguientes líneas.

```
LoadFile /usr/lib/libxml2.so
LoadFile /usr/lib/liblua5.1.so
LoadModule security2_module /usr/lib/apache2/modules/mod_security2.so
```

Ejecutando ***a2enmod modsecurity*** se cargará el módulo en el siguiente inicio de ***Apache***. Adicionalmente será necesario cargar el módulo ***unique id.***

Comprobando el funcionamiento

Por defecto ***mod security*** tan solo alerta ante posibles ataques, ***URL*** malformadas, etc. Es el comportamiento recomendado inicialmente, ya que podría invalidar por completo las aplicaciones web que aloje ***Apache.*** La directiva que se encarga de esta opción en el fichero ***/etc/apache2/mods-available/modsecurity*** es.

```
SecRuleEngine DetectionOnly
```

Un ejemplo del funcionamiento sería el siguiente, en el que se muestra un intento de ***SQLi*** hacia un recurso web. El fichero por defecto donde se registrarán estos eventos es ***/var/log/modsec_audit.log.***

```
Message: Warning. Pattern match "(?i:\\bor\\b ?(?:\\d{1,10}|[\\'\"][^=]
{1,10}[\\'\"]) ?[=<>]+|(?i:'\\s+x?or\\s+.{1,20}[+\\-!<>=])|\\b(?i:x?or)\\b\\
s+(\\d{1,10}|'[^=]{1,10}')|\\b(?i:x?or)\\b\\s+(\\d{1,10}|'[^=]
{1,10}')\\s*?[=<>])" at ARGS:id. [file "/etc/apache2/modsec/base_rules/
modsecurity_crs_41_sql_injection_attacks.conf"] [line "133"] [id "959071"] [rev
"2"] [msg "SQL Injection Attack"] [data "Matched Data: ' or 1=1-- found within
ARGS:id: ' or 1=1--"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.7"] [maturity
"9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/
WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"]
```

Es muy recomendable leer en profundidad la documentación de ***modjsecurity*** para conocer sus funcionalidades y modificar las configuraciones según sea necesario. La página oficial del proyecto es http://www.modsecurity.org**.**

HTTPS

Una de las medidas más implementadas en servidores web es sin duda la utilización de ***HTTP*** sobre ***SSL/TLS.*** En definitiva, el uso de ***HTTPS*** protegerá el tráfico intercambiado entre clientes y el propio servidor web. ***Apache*** soporta esta funcionalidad utilizando el módulo ***mod ssl***, que en el caso de Debian está disponible por defecto con los paquetes instalados inicialmente.

Independientemente del servidor web utilizado será necesario poseer un certificado en dicho servidor con la finalidad de establecer un canal cifrado y por lo tanto seguro. Es posible utilizar certificados firmados por una autoridad certificadora o bien certificados autofirmados. En cualquiera de los casos, las directivas obligatorias a establecer en la configuración del sitio de ***Apache*** son las siguientes.

```
SSLEngine      on                  # Activar / desactivar SSL en el sitio
SSLCertificateFile     <ruta>      # Ruta hacia el fichero del certificado
SSLCertificateKeyFile  <ruta>      # Ruta hacia la clave privada
```

Un ejemplo de configuración ***SSL*** está definida en el fichero ***/etc/apache2/sites-available/default-ssl.*** Dicha configuración es la que se utilizará a continuación para crear un sitio seguro.

Configurando Apache

Para conseguir un entorno ***HTTPS*** básico se realizarán los siguientes pasos en el servidor que ejecuta ***Apache.***

1. Crear un certificado autofirmado con la ayuda de la herramienta ***make-ssl-cert.***
2. Cargar ***modssl*** y el sitio web ***default-ssl*** incorporado por defecto en la instalación de ***Apache.***
3. Cargar el módulo ***mod rewrite*** de ***Apache*** para redirigir las solicitudes de la página ***HTTP*** a su versión ***HTTPS.***

La aplicación ***make-ssl-cert*** facilitará la creación de certificados autofirmados de servidor con el nombre de la máquina en el que se ejecuta la utilidad. Esta utilidad está disponible mediante la instalación del paquete ***ssl-cert.*** Una vez instalada se crea el certificado del siguiente modo.

```
# make-ssl-cert generate-default-snakeoil --force-overwrite
```

El comando anterior habrá generado o sobrescrito los ficheros ***/etc/ssl/certs/ssl-cert-snakeoil.pem*** y ***tetc/ssl/private/ssl-cert-snakeoil.key***, correspondientes al certificado y a su clave privada.
Para activar tanto ***mod ssl*** como el sitio ***default-ssl*** se utilizarán los ya conocidos ***a2enmod***y ***a2ensite.***

```
# a2enmod ssl
# a2ensite default-ssl
```

El sitio ***default-ssl*** estará utilizando el certificado anterior, pues inicialmente dispone de las líneas de configuración:

```
SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
```

En último lugar, si se desea que los clientes accedan de forma automática a la versión segura de la página web, ***Apache*** deberá redirigirlos de forma automática. Para ello debe activarse el módulo

mod rewrite y agregar unas líneas al fichero de configuración de la versión ***HTTP*** de la página. En este caso se ha utilizado la configuración por defecto ***/etc/apache2/sites-available/default***, en la que se han agregado el siguiente contenido.

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        RewriteEngine On
        RewriteCond %{SERVER_PORT} !^443$
        RewriteRule ^/(.*) https://192.168.87.55/$1 [L,R]
...
```

La regla indica una redirección a la dirección IP ***192.168.87.55*** utilizando el protocolo ***HTTPS.*** En este caso se trata de la dirección IP asignada al servidor web ***Apache*** para esta prueba.

Analizando una captura de datos mediante ***Wireshark*** se obtendría el siguiente resultado al acceder al recurso ***HTTP.***

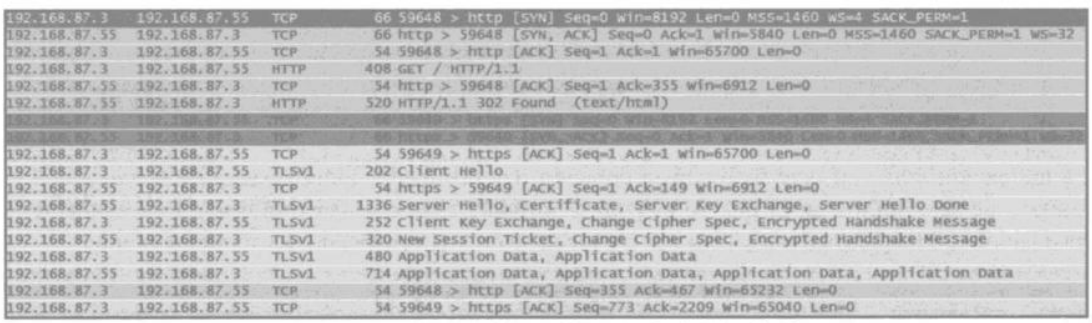


Fig 6.04: *Apache*, *HTTPS* y *mod_rewrite* desde *Wireshark*.

La configuración funciona de acuerdo a lo establecido, por lo que todo indica que los clientes que accedan al recurso estarán protegidos bajo un canal seguro. Sin embargo, ¿realmente siempre se está seguro? La respuesta es no.

SSL-Strip

Como su nombre indica, ***SSL-Strip*** trata de eliminar la capa ***SSL*** en conexiones ***HTTPS***, por lo que al final dicha conexión solo será ***HTTP.*** Es decir, intenta degradar una conexión segura a no segura. Para que un atacante pueda desplegar este ataque deben cumplirse los siguientes requisitos.

- Red o máquinas sin protección ante técnicas ***MiTM*** como ***ARP Spoofing.***
- Redirecciones a sitios seguros ***HTTPS***, desde sitios no seguros ***HTTP.*** Por ejemplo cuando ***Apache*** realiza una redirección con ***mod rewrite*** o bien cuando existe un enlace a un sitio seguro ***HTTPS*** dentro de una web no segura. Este caso suele darse en numerosas ocasiones ante procesos de autenticación o login en páginas web, al pasar de ***HTTP*** a ***HTTPS.***

Por la parte de los clientes casi en la totalidad de las ocasiones se utilizará una dirección no segura ***HTTP*** y se dejará que el servidor actúe redirigiendo a la versión segura ***HTTPS.*** Un ejemplo claro se daría al acceder a www.gmail.com**.**

Continuando con el entorno de redirección del ejemplo se propone un entorno envenenado utilizando ***SSL-Strip.*** La idea consiste en que un usuario atacante intercepte el tráfico entre el servidor web y el cliente o víctima. Para ello se utilizarán los conocidas herramientas ***iptables*** y ***arpspoof*** además de la ya mencionada ***SSL-Strip.***

El atacante realizaría los siguientes pasos en su equipo.

```
# echo "Preparando el entorno MiTM"
# echo 1 > /proc/sys/net/ipv4/ip_forward
# arpspoof -i eth0 -t 192.168.87.55 192.168.87.3 &>/dev/null &
# arpspoof -i eth0 -t 192.168.87.3 192.168.87.55 &>/dev/null &
# echo "Creando redirección para SSL-Strip y ejecutando la aplicación"
# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j \
REDIRECT --to-port 10000
# python /pentest/web/sslstrip/sslstrip.py -w /root/Desktop/dump.txt
```

Es decir el atacante se interpone entre ambas máquinas, el servidor web y el cliente. Nótese que la dirección IP del servidor es ***192.168.87.55*** y la del cliente ***192.168.87.3.*** Por otra parte, todo el tráfico que llegue con destino el puerto ***80/tcp*** será reenviado a la propia máquina del atacante hacia el puerto ***10000/tcp***, que es donde escucha por defecto ***SSL-Strip.*** Realmente se está haciendo que ***SSL-Strip*** sea el nuevo cliente para suplantar la conexión ***HTTPS.*** Por lo tanto al cliente le seguirá apareciendo que la conexión es no segura ***HTTP.***

El atacante podrá observar el fichero de volcado de datos establecido al lanzar ***SSL-Strip*** del siguiente modo, en el que se detectarán campos enviados por ***POST*** como el caso de las credenciales del usuario ***carlos.***

```
# tail -f /root/Desktop/dump.txt
2013-03-04 20:48:45,515 POST Data (192.168.87.55):
usuario=carlos&password=P4ssW0rd%21
```

Medidas de protección

Por parte de los usuarios deberían tomarse las siguientes precauciones.

- Introducción de la URL con ***https://*** siempre que sea posible.
- Uso de ***plugins*** del navegador para forzar una conexión ***HTTPS*** desde el inicio.
- Protecciones ante ***ARP Spoofing***, ***IP Spoofing***, etc.
- Verificar la información del navegador indicando si la navegación es segura.

En la parte de la infraestructura de red, programación de los sitios web y configuración de ***Apache.***

- Prevención de ataques ***MiTM*** en la red local mediante la electrónica de red.
- Evitar redirecciones desde páginas no seguras a páginas seguras.
- Programación de páginas web con redirecciones a nivel de cliente, por ejemplo mediante ***JavaScript.***
- Utilización del mecanismo ***HSTS***, ***HTTP Strict-Transport-Security*** en ***Apache.*** Se trata de una cabecera para utilizar junto con ***mod headers*** con la que se podrá indicar a los clientes que ese recurso se ofrece de forma segura; y que por lo tanto sea el propio ***UserAgent*** del cliente el que inicie la conexión ***HTTPS*** de forma automática. Lógicamente los navegadores web deben soportar este mecanismo. Se puede encontrar más información sobre esta funcionalidad en el ***RFC 6797.***

Un ejemplo de ***HSTS*** en la cabecera de una respuesta del servidor web ***Apache*** sería el siguiente.

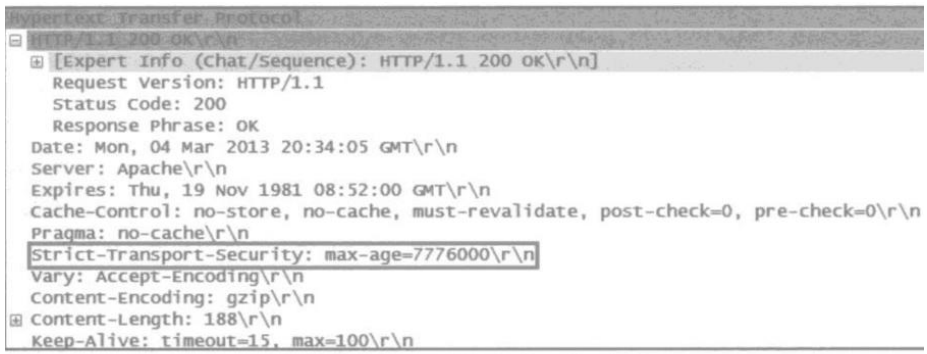


Fig 6.05: Cabecera de *HSTS* en *Wireshark.*

Además de las medidas ya comentadas para Apache, se trata de un servicio muy configurable en cuanto a rendimiento se refiere. Bajo entornos de mucha carga es indispensable conocer los parámetros de configuración para afinar el rendimiento del servidor web. Como se comentaba al inicio de este capítulo, desde las páginas oficiales se encontrará una amplia documentación. Por otra parte se recomienda la visita al sitio www.owasp.org, donde se encuentra un gran repertorio de artículos, herramientas, foros, listas de correo, etc. sobre seguridad.

Capítulo VII
Fortificación y seguridad en SSH

1. Introducción a SSH

SSH, ***Secure SHell,*** es uno de los protocolos más interesantes que se dispone en la administración de sistemas, ya que permite acceder a otros sistemas de manera remota, y además de forma segura. SSH es el nombre que recibe tanto el protocolo como la aplicación que lo implementa. Su principal función es acceder a máquinas remotas a través de las redes, proporcionando al administrador una gestión completa de los equipos remotos mediante los túneles SSH, es decir, seguros.

Las distintas funcionalidades que proporciona el protocolo hacen que este sea fundamental para los administradores de hoy en día. Con SSH se puede utilizar un intérprete de comandos de forma remota, copiar volúmenes de datos, disponer de sesiones FTP seguras, utilización de claves RSApara fortificar la autenticación y creación de los canales seguros, crear túneles inversos o utilizar estos para fortificar otras aplicaciones mediante la potencia de este protocolo, etcétera. Históricamente se ha relacionado a SSH con el protocolo Telnet, ya que este trabaja de forma similar, con la gran diferencia de que SSH utiliza técnicas de cifrado para asegurar el envío de la información entre los sistemas. Esta información viaja por unos túneles cifrados lo cual hace que la información sea no legible por posibles atacantes. Existe la posibilidad de atacar este tipo de sistemas, en algunas versiones antiguas, por medio de ataques de ***replay*** o reinyección, y manipular de este modo la información entre los destinos.

SSH utiliza un intercambio de claves basado en el protocolo criptográfico de ***Dijfie-Hellman.*** Este protocolo es imprescindible para la construcción de la capa de transporte segura. SSH es un protocolo que proporciona una capa importante de seguridad para los administradores, pero en función de la configuración del servicio puede llevar al sistema a un estado de inseguridad. Es importante entender el funcionamiento del protocolo, las funcionalidades que presenta y la configuración óptima en función de las necesidades del administrador o la empresa.

Funcionamiento del protocolo

En la imagen se puede visualizar como en primer lugar se realiza la conexión mediante TCP, realizando el ***three-way handshake*** o triple apretón de manos (,***SYN***, ***SYN+ACK, ACK).*** Después

de abrir la conexión, cliente y servidor se envían la versión disponible del protocolo. Actualmente existen dos versiones del protocolo SSH, la versión 1 y la 2. La versión número 1 es totalmente desaconsejable, ya que se han encontrado fallos de seguridad. Después de enviarse la versión del protocolo disponible, se utilizará la clave pública y privada de RSA, ***Rivest Shamir*** y ***Adleman.*** El servidor enviará su clave pública de ***host*** al cliente para que este pueda cifrar lo que necesite enviar al servidor en un instante posterior. El cliente comparará la clave pública de ***host*** con la que tenga almacenada del servidor en archivo ***knownjiosts***, en la ruta ***$HOME/.ssh.*** Si es la primera vez que se realiza la conexión se mostrará una alerta donde se indica que no se puede verificar la identidad del servidor por parte del cliente, o si por el contrario, la clave pública es distinta a la que se tiene almacenada por el cliente se mostrará una alerta indicando que la clave ha cambiado. De este modo se alerta al cliente que el servidor no dispone de la misma clave. La manera óptima de enviar, por primera vez, la clave pública de ***host*** a un cliente por parte de un servidor es vía física, aunque en algunos casos no se puede realizar de este modo, por lo que se debe confiar en el peligro. Otra opción es identificar el ***fingerprint*** o huella para verificarlo vía telefónica con el administrador del servidor.

Fig 7.01: Establecimiento de la conexión mediante el protocolo SSH.

Una vez el cliente dispone de la clave pública de ***host*** del servidor, este generará una clave de sesión aleatoria y seleccionará un algoritmo de cifrado simétrico. Con esta clave de sesión, la cual por defecto se regenerará cada 3600 segundos, se cifrará el túnel. El cliente enviará un mensaje conteniendo la clave de sesión y el algoritmo de cifrado seleccionado, esta información viajará cifrada con la clave pública de ***host*** al servidor usando el algoritmo RSA. En este instante, el resto de la comunicación se utilizará el algoritmo de cifrado simétrico y la clave compartida de sesión, ya que este método es más rápido para cifrar y descifrar, pero debe ser usado primero el protocolo de ***Diffie-Hellman*** para poder enviar de manera segura la clave compartida de sesión.

Una vez ya se encuentra el túnel seguro creado se llevará a cabo la autenticación del usuario en el servidor. Se puede utilizar distintos métodos para llevar a cabo la autenticación.

La primera conexión

Antes de realizar la primera conexión básica a un servidor mediante el protocolo SSH se especifica cómo realizar la instalación de un servidor SSH. Para instalar el servidor en una distribución de ***GNU/Linux***, por ejemplo una ***Debían***, se utilizará aplicaciones como ***apt-get*** o ***aptitude.*** Se podría utilizar otros instaladores como ***YaST*** o ***Synaptic*** para otras distribuciones. La instrucción a ejecutar es ***[sudo] apt-get install openssh-server*** y la configuración del servicio se llevará a cabo en los archivos alojados en la ruta ***/etc/ssh.***

```
pablo@pablo-VirtualBox:~$ sudo apt-get install openssh-server
Leyendo listas de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes extras:
```

Fig 7.02: Instalación del servidor SSH.

Para poder iniciar una sesión remota se debe disponer de una cuenta en el sistema con el que se pretende conectar. Para realizar la conexión se ejecutará la siguiente sintaxis ***ssh usuario@ip.*** El valor de IP puede ser sustituido por un nombre de dominio.

Por defecto un servicio SSH se está ejecutando en el puerto 22 de una máquina. Si el administrador cambia el valor por defecto a otro puerto y se quiere iniciar una conexión se deberá activar el ***jiag*** -p de la siguiente manera ***ssh -p <puerto> usuario@ip.***

Como nota rápida indicar que se pueden realizar sesiones gráficas mediante el protocolo SSH, cuyo tema se entrará en detalle más adelante. Para activar este tipo de sesiones se ejecuta la instrucción ***ssh -X usuario@ip***, cabe destacar que si el puerto de conexión fuera distinto al de por defecto, se debería indicar el ***flag*** -p como se mencionó anteriormente. A priori, puede parecer que la sesión es idéntica a la abierta con las instrucciones anteriores, pero si se ejecuta alguna herramienta gráfica se obtiene la imagen de esta en remoto.

2. Configuración del servicio

Existen, principalmente, dos conjuntos de archivos de configuración. Todos se encuentran en la ruta /***etc/ssh***, pero el primer conjunto corresponde con la configuración de las aplicaciones cliente como son ***ssh, scp*** o ***sftp***, y el segundo conjunto corresponde con la configuración del demonio ***sshd.***

El servidor o demonio dispone de un archivo de configuración principal el cual es ***/etc/ssh/sshd_ config.*** El cliente dispone de un archivo de configuración el cual se encuentra en ***/etc/ssh/ssh config.*** Es importante notar la diferencia entre uno y otro, ya que se diferencian en una sola letra. Hay que destacar que en el caso del cliente, si un usuario dispone de un archivo de configuración en su ***home***, como por ejemplo ***$HOME/.ssh/ssh_config***, será este último archivo del que se cargue la configuración para el cliente y no la que se encuentra en la ruta ***/etc/ssh/ssh config.***

En este apartado se estudiarán distintos archivos de interés y un alto número de directivas y el significado de estas. Además, se profundizará en el uso de las autenticaciones mediante contraseña o clave pública.

Archivos del servicio

Antes de entrar en detalle en los archivos de configuración del servicio, sobretodo ***sshd config***, se resume una serie de archivos que deben conocerse para el correcto funcionamiento y configuración del servicio.

El primer archivo a tener en cuenta es ***moduli***, situado en la ruta ***/etc/ssh.*** Contiene grupos de ***Diffie-Hellman*** los cuales son usados para el intercambio de claves, como se mencionó anteriormente, cuyo procedimiento es imprescindible para la creación de una capa segura de transporte.

Los siguientes archivos referencian a las claves públicas y privadas de ***host***, las cuales son utilizadas para el intercambio de claves. Estas claves de ***host*** se encuentran en el directorio ***/etc/ssh.***

- ***ssh host dsa key*** y ***ssh host dsa key.pub.*** El archivo con extensión ***pub*** se refiere a la clave pública y el archivo sin extensión es la clave privada de ***host.*** Se utiliza DSA por el demonio ***sshd.***
- ***ssh host rsa key*** y ***ssh host rsa key.pub.*** Clave pública y privada de RSA utilizada por el demonio ***sshd*** en su versión 2 o SSHv2.
- ***ssh host key*** y ***ssh host_key.pub.*** Clave pública y privada utilizada por el demonio ***sshd*** en su versión 1 o SSHvl.

Cada usuario dispone en su directorio ***$HOME/.ssh*** de una serie de archivos que, después de generarse en algunos casos, se deben conocer. A continuación se especifican en el siguiente listado:

- ***Authorizedkeys.*** Este archivo contiene un listado de claves públicas, para la autenticación contra un servidor de un cliente. Este listado refleja las claves públicas autorizadas por el usuario para ser usadas en su cuenta. Cuando un cliente se conecta a un servidor mediante SSH, y la autenticación elegida es mediante clave pública y privada, se chequea el valor de la clave pública en este listado, si el cliente dispone de la clave privada se autenticará en el sistema.
- ***Known hosts.*** Este archivo contiene las claves públicas de ***host*** de los servidores a los que se ha conectado un cliente. Este fichero es importante para asegurarse de que el cliente SSH se está conectado al servidor correcto y no a uno suplantado.
- ***id dsa*** e ***iddsa.pub.*** Clave privada y pública DSA de usuario.
- ***id rsa*** e ***id rsa.pub.*** Clave privada y púbica RSA de usuario para la versión SSHv2.
- ***identity*** e ***identity.pub.*** Clave privada y pública RSA de usuario para SSHvl.

Tras el repaso a archivos importantes que se irán utilizando o configurando más adelante, se debe especificar los archivos de configuración de cliente y servidor. En primer lugar se trata el fichero

sshd_config, el cual permite configurar el comportamiento del servidor SSH. Este archivo se puede editar mediante un editor de textos y su formato es realmente sencillo. El formato del fichero se divide en líneas, cada línea indica una directiva que implica un comportamiento en el servicio. Como se puede visualizar en la imagen se observa una serie de directivas y los valores asociados a estas en la parte derecha. Hay que recordar que las líneas que comienzan por el símbolo *#' no se encuentran en ejecución.

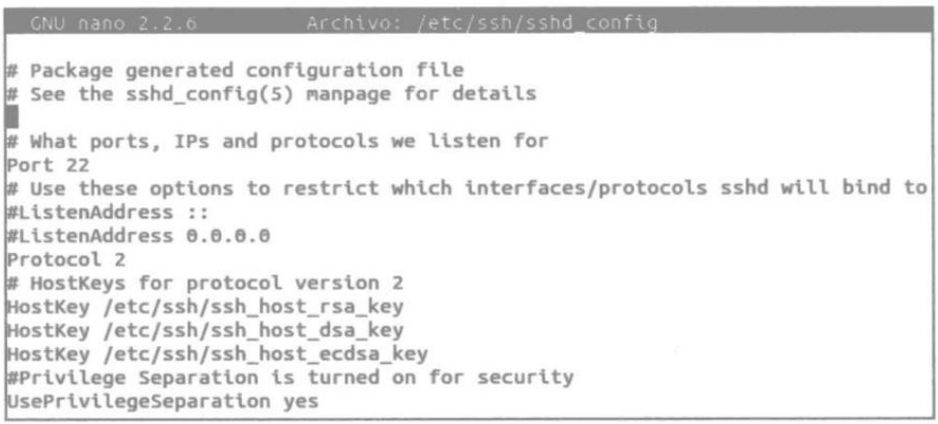


Fig 7.03: Archivo de configuración de *sshd*.

El fichero de configuración del cliente es similar al del servidor, aunque dispone de otras directivas. La versión del protocolo, el puerto de conexión por defecto, el listado de algoritmos de cifrado disponible, entre otras opciones se pueden configurar en este archivo ***/etc/ssh/ssh_config.***

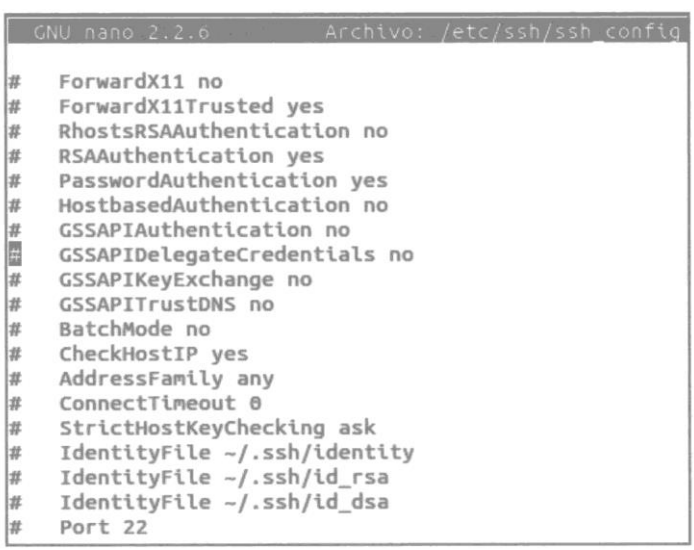


Fig 7.04: Archivo de configuración del cliente *ssh*.

Directivas básicas

En este apartado se especifican las directivas básicas para la configuración y mejora de seguridad en el servidor SSH. Además, se exponen algunas directivas interesantes para forzar una mejora de seguridad en el cliente SSH. El protagonista de esta sección son los archivos de configuración ***sshd_config*** y ***ssh_config.*** Existe un gran número de directivas que pueden resultar interesantes para el administrador aparte de las que se enumeran en este apartado.

En primer lugar se trata el fichero ***sshdconfig*** con este listado de directivas cuya configuración puede provocar la mejora de seguridad en el servicio:

- ***Port.*** La directiva***port*** indica en que puerto se colocará a la escucha el servicio SSH. Por defecto está configurado en el puerto 22, pero se puede cambiar ese valor por otro cualquiera. Hay que tener en cuenta que si se comenta la directiva se utilizará el valor por defecto, es decir, el 22. Si el servicio SSH es utilizado en una interna puede ser realmente interesante colocar el servicio a la escucha en un puerto alto para evitar escaneos simples por parte de un potencial atacante. Si el servicio es público, por norma general, deberá quedarse en el puerto 22.

- ***PermiíRootLogin.*** Con esta directiva se prohíbe que un usuario se loguee en el servidor con el usuario ***root.*** De este modo se evita ataques de fuerza bruta al usuario ***root***, o por si por otra vía se consigue la clave de ***root*** no se podrá acceder al sistema vía SSH. De este modo un usuario que sea administrador deberá, en primer lugar, loguearse como su usuario y después cambiar a ***root*** o utilizar ***sudoer.***

- ***MaxAuthTries.*** Esta directiva evita que los ataques de fuerza bruta puedan estar probando indefinidamente credenciales. Limitar los intentos a un número bajo ayuda a evitar lo comentado anteriormente. Si una vez superado el número de intentos que indica la directiva no se ha logueado en el sistema se abortará la conexión. Para habilitar la directiva se debe especificar la siguiente línea ***MaxAuthTries <número intentos>.***

- ***LoginGraceTime.*** Esta directiva indica el tiempo máximo, en segundos, para introducir las credenciales en la autenticación. Es aconsejable utilizar un tiempo bajo, en tomo a los 30 o 45 segundos, en función de los elementos de seguridad que haya entre el cliente y el servidor. La sintaxis de la directiva es ***LoginGraceTime <valor segundos>.***

- ***AllowGroups.*** Esta directiva especifica el nombre de los gmpos a los que pertenecen los usuarios que pueden iniciar sesión de manera remota mediante el protocolo SSH. En un entorno corporativo puede ser interesante decidir que departamentos pueden iniciar sesión o qué gmpo de usuarios con privilegios pueden llevar a cabo dicha acción. Un ejemplo sería habilitar la directiva ***AllowGroups*** y definir un gmpo denominado ***UsuariosRemotos***, de tal forma que solo los usuarios que pertenezcan a dicho gmpo podrán iniciar sesión remota. En el momento de habilitar dicha directiva se restringe el acceso a todos los usuarios que no se encuentren en alguno de los gmpos indicados en el valor de la directiva. Los gmpos se deben separar por espacios en blanco del siguiente modo ***AllowGroups <grupol> <grupo 2> ... <grupo N>.***

- ***AllowUsers.*** Esta directiva es similar a la anterior, se puede especificar en el fichero de configuración seguida de una lista de usuarios que pueden iniciar sesión mediante el protocolo SSH. Se puede usar metacaracteres, como o '?\ lo cual permite la construcción de patrones. Por ejemplo, si el patrón sigue la forma usuario@host, entonces el servicio SSH comprobará por separado tanto el usuario como el host del que proviene la solicitud, de este modo se consigue atar un usuario a un host o dirección IP concreta.
- ***Ciphers.*** Esta directiva especifica que cifrados admitirá la versión del protocolo. Si se especifican varios algoritmos, estos deben ser separados por comas. Un ejemplo sería ***aesl28-cbc,3des-cbc,blowfish-cbs,castl28-cbc,arcfour***\
- ***ClientAlivelnterval.*** Habilitar esta directiva suponer que si el servidor no recibe datos del cliente en un tiempo X, cuyo valor se especifica junto a la directiva, el servidor enviará un mensaje a través del canal solicitando respuesta al cliente. Su sintaxis es ***ClientAlivelnterval <tiempo segundos>.***
- ***ClientAliveCountMax.*** Esta directiva está relacionada con la anterior. Se especificará el número de mensajes que el servidor puede enviar al cliente solicitando respuesta antes de cerrar la conexión, si este no contesta.
- ***TCPKeepAlive.*** Deshabilitar esta directiva permite prevenir ataques de suplantación, ataques de tipo ***spoofing.***
- ***DenyGroups.*** Esta directiva es similar a la de ***AllowGroups*** pero con un enfoque de denegación, es decir, todo grupo que se encuentre asociado a dicha directiva no podrá iniciar sesión. Generalmente, es más utilizada la directiva ***AllowGroups***, ya que es restrictiva por omisión.
- ***DenyUsers.*** Contraria a la directiva ***AllowUsers***, todo usuario que se encuentre asociado a esta directiva no podrá iniciar sesión de manera remota mediante el protocolo SSH. Incluso con el uso de los metacaracteres se podría indicar usuarios que no pueden conectarse desde algunos ***hosts*** concretos.
- ***LogLevel.*** Esta directiva es interesante para depurar y registrar las acciones que ocurren en el servicio. Los posibles valores son ***QUIET, FATAL, ERROR, INFO, YERBOSE*** y ***DEBUG.*** El predeterminado es ***INFO*** y se ha de tener cuidado con ***DEBUG*** ya que viola la privacidad de los usuarios, por lo que no es recomendable en un entorno de producción.
- ***HostKey.*** Esta directiva indica la ruta donde se encuentra la clave privada RSA o DSA del ***host.*** Generalmente, las claves de ***host*** son generadas en el momento de la instalación del servidor SSH. Puede ser recomendable realizar una regeneración de clave de ***host*** con mayor número de bits.

UsePrivilegeSeparation. Esta directiva permite la separación de privilegios activada, es decir, divide los procesos del servidor para prevenir la explotación y la posible escalada de privilegios.

- ***Protocol.*** Esta directiva indica la versión del protocolo que se utilizará. Es importante que se encuentre solo activada la versión 2 del protocolo, por lo que se deberá configurar de la siguiente manera ***protocol 2.*** Existen configuraciones que reflejan la siguiente directiva

protocol 2,1, indicando que si el cliente no dispone de la versión 2 se utilizará la 1. Esto no es nada recomendable, por lo que si el cliente no dispone de la versión 2, no se debe dejar conectar con el servidor hasta que no actualice su configuración.

- ***PubkeyAuthentication.*** Si esta directiva se encuentra con valor ***yes,*** entonces el servidor permitirá la autenticación de usuarios mediante el uso de clave pública. En entornos críticos es altamente recomendable habilitar este tipo de autenticación y deshabilitar otros métodos menos seguros.
- ***AuthorizedKeysFile.*** Esta directiva indica al servidor donde se encuentran almacenadas las claves públicas de los usuarios. Por defecto, las claves se almacenan en ***$HOME/.ssh/ authorized keys,*** es decir en ***%h/.ssh/authorized_keys***, pudiendo ser cambiada esta ruta en el valor de esta directiva.
- ***PasswordAuthentication.*** Si esta directiva se encuentra con valor ***yes***, entonces el servidor permitirá la autenticación de usuarios mediante contraseñas. Esta directiva no es recomendable para entornos críticos, por lo que se debería configurar con valor ***no.***
- ***XllForwarding.*** Esta directiva indica si se permite el reenvío de XI1 a través de la conexión SSH. Es cuestión del administrador y las necesidades de los usuarios el habilitar o no esta directiva. Si se sabe de antemano que no se utilizará un entorno gráfico, será una buena opción deshabilitar este reenvío.
- ***ListenAddress.*** Esta directiva indica por cual dirección se debe escuchar las peticiones. Puede que un servidor donde se encuentre instalado el servicio SSH disponga de varias tarjetas de red, pero por seguridad solo se requiere que sea la tarjeta de red que comunica el servidor con una red interna. Por ejemplo, se dispone de un servidor con tres tarjetas de red, una proporciona salida hacia Internet mientras que las otras dos son redes internas de la empresa. Puede ser que por seguridad se requiera que solamente se acceda a través de una tarjeta de red interna al servicio SSH, por lo que con esta directiva se indica que dirección IP o red está permitida. Su sintaxis es ***ListenAddress <red 1> ... <redN>.***
- ***ServerKeyBits.*** Esta directiva indica el tamaño de la clave de sesión. Es altamente recomendable utilizar un tamaño de clave mayor de 768 bits, que es el valor que se dispone en gran cantidad de versiones de la aplicación. Se debería cambiar por un valor como 1024 o 2048 bits.
- ***KeyRegenerationlnterval.*** Esta directiva indica el tiempo, en segundos, que transcurrirá hasta que se genere una nueva clave de sesión. Por defecto, se utiliza un valor de 3600 segundos, es decir, cada hora. Este valor es bastante seguro, pero si se requiere de una mayor seguridad, incluso se puede bajar.
- ***Banner.*** Con esta directiva se muestra un texto de advertencia, por ejemplo, exponiendo temas legales sobre un posible acceso no autorizado. Esta directiva tiene un valor que indica la ruta donde se encuentra escrito dicho texto, por defecto es la ruta ***/etc/issue.net.***
- ***PrintLastLog.*** Con esta directiva habilitada se registra el último inicio de sesión del usuario.

Una vez las directivas son configuradas se debe reiniciar el servicio mediante la instrucción ***Service ssh restart*** o ***/etc/init.d/ssh restart.*** A continuación se especifican directivas del cliente SSH que pueden resultar de interés para mejorar la fortificación del servicio por parte del cliente.

- ***Host.*** Esta directiva actúa como un divisor de secciones. Puede repetirse en varias ocasiones en el archivo ***sshjconfig***, y su valor, el cual puede ser un conjunto de patrones, determinará que las directivas de dicha sección son aplicadas a las conexiones realizadas a los ***host*** en cuestión. El valor o metacaracter '*' indica que las directivas y sus valores se aplicarán a todos los ***hosts*** o conexiones a servidores.
- ***Port.*** Esta directiva es similar que en la configuración del servidor, salvo que ahora indica a que puerto se conectará el cliente por defecto.
- ***Protocol.*** Esta directiva indica la versión del protocolo que utilizará el cliente. Como se mencionó en la configuración del servidor es altamente recomendable utilizar, solamente, la versión 2 del protocolo.
- ***IdentifyFile.*** Esta directiva que indica donde se encuentra almacenada la clave, es decir, se indica la ruta y el fichero donde se encuentra, por ejemplo, ***$HOME/.ssh/id_rsa.***
- ***PubkeyAuthentication.*** Indica al cliente si se puede autenticar contra el servidor mediante autenticación de clave pública.
- ***StrictHostKeyChecking.*** Esta directiva define que realizará el cliente al conectarse con el servidor del cual no se dispone su clave pública de ***host.*** El valor ***yes*** hace que se acepte automáticamente la clave recibida, el valor ***no*** hace que se rechace la clave del servidor, lo que significa que se cierra la conexión, mientras que el valor ***ask*** hace que se pida confirmación al usuario.
- ***Ciphers.*** Lista de algoritmos de cifrado simétrico que puede utilizar el cliente.
- ***PasswordAuthentication.*** Indica si el cliente puede autenticarse mediante contraseña contra el servidor.

Autenticación con contraseña

Existen distintos métodos para autenticar a usuarios en el servidor SSH. El método más básico y más utilizado por la mayoría de los usuarios en su día a día es la autenticación por contraseña. Cuando el cliente se conecte al servidor, este le pedirá que introduzca la contraseña para el usuario con el que se quiere iniciar sesión. Este tipo de autenticación es denominada básica, ya que es la más común.

Existen ciertos entornos en el que este tipo de autenticación no debe ser permitido por varias razones de seguridad. Esta autenticación está considerada como uno de los niveles más bajos para autenticar usuarios. Deshabilitar este tipo de autenticación mediante la directiva ***PasswordAuthentication*** es altamente recomendable para entornos críticos donde la autenticidad de los usuarios sea algo crítico.

Este tipo de autenticación tiene algunos problemas de seguridad como puede ser un ataque de fuerza bruta, ataque de diccionario, usuarios que configuren contraseñas no seguras o no complejas, usuarios que dan su contraseña a otros, usuarios que disponen de un ***keylogger*** en su equipo, etcétera.

Antes de explicar la autenticación mediante clave pública, se debe repasar la autenticación basada en ***host/user.*** Este tipo de autenticación indica que usuario y desde que máquina se permite el acceso al servidor.

```
pablo@pablo-VirtualBox:/etc/ssh$ ssh pablo@192.168.1.35
Bienvenido al servicio SSH
Informatica 64
Pablo Gonzalez
pablo@192.168.1.35's password:
```

Fig 7.05: Autenticación por contraseña.

Clave pública y clave privada

Este tipo de autenticación es una de las más seguras a la hora de configurar el acceso de los usuarios y permite obtener el máximo partido al servicio SSH. El cliente debe generar su par de claves, privada y pública, compartiendo la clave pública con el servidor al que se quiere tener acceso mediante este tipo de autenticación. Hay que recordar que un mensaje cifrado mediante clave pública solo podrá ser descifrado con su clave privada.

El cliente solicitará autenticarse en el servidor y este le mostrará los distintos métodos que hay disponibles, según la configuración del ***sshd.*** Si la autenticación por clave pública se encuentra habilitada el servidor cifrará un mensaje con la clave pública del usuario que requiere el acceso, si este es capaz de contestar al desafío estará demostrando al servidor que es quién dice ser, o al menos que está en posesión de la clave privada que contesta al desafío del servidor.

La clave privada puede tener un segundo mecanismo de seguridad, autenticación multifactor, denominado ***PIN.*** Cada vez que se requiera el uso de la clave privada para descifrar el desafío del servidor se solicitará al usuario que posee la clave privada que introduzca el ***PIN*** para verificar que posee dicha clave y conoce su ***PIN*** para poder utilizarla. En algunas ocasiones interesa automatizar el proceso de autenticación por lo que la clave privada de un usuario se genera sin ***PIN***., para que se puedan escribir ***Scripts*** que utilicen estos mecanismos sin que se solicite ningún dato a los usuarios, por ejemplo, la automatización de un ***backup*** de usuario a otro servidor destino.

Generación y autenticación con clave pública

Para llevar a cabo el proceso de generación del par de claves se utilizará la herramienta ***ssh-kéygen.*** Si se ejecuta en una terminal la instrucción ***ssh-keygen*** sin ningún parámetro se creará un par de claves por defecto. Estas claves se almacenarán por defecto en el ***home*** del usuario, en la carpeta oculta ***.ssh.***

Es realmente interesante utilizar el comando ***man*** para obtener un listado de parámetros de esta herramienta y ver las posibilidades que esta ofrece. Para el ejemplo se generará un par de claves de tipo RSA con 4096 bits. Para ello se ejecutará la instrucción ***ssh-keygen -b 4096 -t rsa.*** Se indica el

directorio por defecto donde se almacenarán las claves y si se quiere incluir una contraseña o ***PIN*** para proteger el uso de la clave privada.

```
pablo@pablo-VirtualBox:~$ ssh-keygen -b 4096 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pablo/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pablo/.ssh/id_rsa.
Your public key has been saved in /home/pablo/.ssh/id_rsa.pub.
The key fingerprint is:
08:f3:29:7a:fe:51:c7:17:7e:65:9c:16:c9:60:a6:26 pablo@pablo-VirtualBox
The key's randomart image is:
+--[ RSA 4096]----+
|             +o..|
|            +  +o|
|    o    E o.  o+|
|     + o .o. ..o |
|    . + S o o .  |
|   . . . . . .   |
|  . . .          |
|   o   .         |
|    ...          |
+-----------------+
pablo@pablo-VirtualBox:~$
```

Fig 7.06: Generación del par de claves RSA.

Una vez que se dispone de la clave pública y privada para autenticación del usuario creada, se debe copiar la clave pública en el servidor al que se quiere acceder mediante esta autenticación. La clave se subirá al servidor, y en función de la ruta que se indique en el fichero ***/etc/ssh/sshd_config*** en la directiva ***AuthorizedKeysFile.*** Por defecto esta directiva indica que la clave pública autorizada para la autenticación se encuentra en el ***home*** del usuario en la ruta ***$HOME/.ssh/authorized_keys.***

```
pablo@pablo-VirtualBox:~/.ssh$ cat id_rsa.pub >> authorized_keys
pablo@pablo-VirtualBox:~/.ssh$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC0XpcSYTXhTVh7Y/OD95yykAlrsyA+kxNo4deu/hz+
mBkWpZsH+ccHwb4kgJFO5IPmDgRIVJZY9uwc3jaMqSYThJrw4KXKWp+Fn5bRD/7Xvz5HM9yTvZnzS2r1
5bjJQLk+YChJMXs4ZMb+eYOrPeYwPG5PS+oA6jAW/BK/sm8/FRaBpNeehcXaS0MId3irPO8622j3k8yN
dgfw+9hzmcR1bBACFUlVWvYqITPBeV2yTUlbCzi/Oqzxadu1dp5S15psIh11qowND1IyD9LdNTEMh4+0
nCizH/ygB+4Wl7C1uR5bFeT4yuJbj+Ovne071IY3ElA74+m70mlXGOtlG3CceP0urapkpSUxwZOl8MCX
v5PzHeRuDwRefPdk1INlKKZtkHB2NBPjNqRDuSwOxPdV+5IS9EgYYxf3JtfxEFj2STv5nsvyU2pZU9mC
ZgzWmITGwQnIhXv/4c5j2PyHgrT05z1LOELjVjPs26vXR9eO2rAjVlQTn6APBpj9BdxMPAqdIFpaWBh6
8OB1LSLTzKQRXnMIKfA9iQvv6w+6UQvz/n087zM3yaDQ8gE8OvahuiCAw0WjUpx07na70ptPXigjaJLw
6zurQIuLJbpf+c0YJCpUoJT3tFI5WmIAmDRimlP95o7QQJV++C8gtsJrcpY4UUTUJvSuwx4I/vz/O7Nm
1w== pablo@pablo-VirtualBox
```

Fig 7.07: Almacenar la clave pública en el fichero *authorized_keys*.

Tras subir la clave pública al servidor y copiar su contenido al fichero comentado anteriormente ya se dispondrá de la autenticación mediante clave pública. Como se puede visualizar en la imagen el acceso ya se realiza a través de la autenticación mediante este tipo y no se pide contraseña al usuario.

```
pablo@pablo-VirtualBox:~$ ssh pablo@192.168.1.40
Bienvenido al servicio SSH
Informatica 64
Pablo Gonzalez
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)

 * Documentation:  https://help.ubuntu.com/

497 packages can be updated.
233 updates are security updates.

New release '12.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Jan 27 17:35:28 2013 from pablo-virtualbox.local
pablo@pablo-VirtualBox:~$
```

Fig 7.08: Autenticación con clave pública.

¿Dónde almacenar las claves?

La clave pública se puede repartir por todos los servidores o máquinas a los que el usuario quiera acceder mediante este tipo de autenticación. Por el contrario la clave privada debe ser protegida y un fichero con las mínimas copias posibles, es decir, un ***backup*** bajo control y cifrado. La clave privada deberá almacenarse en un medio extraíble, el cual debería estar protegido mediante cifrado, para evitar su acceso si dicho medio es robado o perdido.

Es realmente importante proteger dicha clave privada, ya que si no se le aplica un ***PIN*** porque será utilizada para la automatización de ciertos procesos mediante SSH, al menos que su almacenamiento proteja al fichero. Además, y como se está estudiando en este libro, otros elementos de seguridad son fundamentales para proteger la privacidad de los usuarios. Elementos como ***firewalls***, ***IDS, IPS***, accesos no permitidos, protección física, complementan y fortalecen el modelo de defensa en profundidad o ***Depth In Defense.***

Resumen del proceso de conexión

Mediante la ejecución del parámetro -v en el lanzamiento del comando ***ssh,*** se puede visualizar y verificar todas las acciones que se realizan por parte del servidor y el cliente en una conexión de este tipo.

En el ejemplo de conexión que se muestra a continuación se puede visualizar como el cliente SSH lee el fichero de configuración ***/ect/ssh/ssh_config*** y aplica el perfil * o por defecto. Tras leer este fichero se crea la conexión TCP con la dirección IP donde se encuentra el servidor. Una vez se realiza el ***three-way handshake*** entre el cliente y el servidor la conexión queda establecida, pero aún no existe un canal cifrado.

El cliente verifica la existencia de posibles claves privadas y verificando si se encuentra en alguna lista negra. Una vez realizado este proceso se verifica la versión del protocolo. En este instante se

intercambia la clave pública de ***host*** del servidor al cliente, para que este pueda generar y cifrar la clave simétrica, la cual será la encargada de cifrar el canal.

Una vez el cliente recibe la clave de ***host*** la verifica en el fichero ***known hosts.*** Si se encuentra en este fichero dicha clave se verifica la identidad del servidor, si por el contario no se encuentra en dicho fichero se mostraría un mensaje avisando de una posible suplantación.

```
pablo@pablo-VirtualBox:~$ ssh -v pablo@192.168.1.40
OpenSSH_5.8p1 Debian-7ubuntu1, OpenSSL 1.0.0e 6 Sep 2011
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 192.168.1.40 [192.168.1.40] port 22.
debug1: Connection established.
debug1: identity file /home/pablo/.ssh/id_rsa type 1
debug1: Checking blacklist file /usr/share/ssh/blacklist.RSA-4096
debug1: Checking blacklist file /etc/ssh/blacklist.RSA-4096
debug1: identity file /home/pablo/.ssh/id_rsa-cert type -1
debug1: identity file /home/pablo/.ssh/id_dsa type -1
debug1: identity file /home/pablo/.ssh/id_dsa-cert type -1
debug1: identity file /home/pablo/.ssh/id_ecdsa type -1
debug1: identity file /home/pablo/.ssh/id_ecdsa-cert type -1
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.8p1 Debia
n-7ubuntu1
debug1: match: OpenSSH_5.8p1 Debian-7ubuntu1 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.8p1 Debian-7ubuntu1
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: sending SSH2_MSG_KEX_ECDH_INIT
debug1: expecting SSH2_MSG_KEX_ECDH_REPLY
debug1: Server host key: ECDSA d9:bd:50:29:01:2b:a7:f3:68:4a:6b:29:f6:6f:71:aa
debug1: Host '192.168.1.40' is known and matches the ECDSA host key.
debug1: Found key in /home/pablo/.ssh/known_hosts:3
debug1: ssh_ecdsa_verify: signature correct
```

Fig 7.09: Operaciones en la conexión SSH.

Tras aceptar el cliente al servidor, verificando la identidad de este, se establece la conexión en un canal ya seguro. El servidor envía el ***banner*** o mensaje de inicio al cliente, y este lo muestra por pantalla. El servidor envía los métodos de autenticación para el usuario que se disponen, como por ejemplo, ***public key*** o ***password authentication.*** Tras autenticarse se envían desde el servidor las variables de entorno de la sesión y proporciona el control de la ***shell*** en remoto.

```
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: Roaming not allowed by server
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
Bienvenido al servicio SSH
Informatica 64
Pablo Gonzalez
```

Fig 7.10: Autenticación finalizada en la conexión SSH.

```
Informatica 64
Pablo Gonzalez
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Offering RSA public key: /home/pablo/.ssh/id_rsa
debug1: Server accepts key: pkalg ssh-rsa blen 535
debug1: Authentication succeeded (publickey).
Authenticated to 192.168.1.40 ([192.168.1.40]:22).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = es_ES.UTF-8
Welcome to Ubuntu 11.10 (GNU/Linux 3.0.0-12-generic i686)
```

Fig 7.10: Autenticación finalizada en la conexión SSH (continuación).

3. Aplicaciones con SSH

El protocolo seguro SSH es utilizado por un gran número de aplicaciones para realizar sus acciones a través de la red de manera segura. Las aplicaciones que se estudiarán en este apartado aportan una funcionalidad necesaria para la administración de servidores y sistemas. El protocolo SSH aporta una capa de seguridad a estas funciones, las cuales realizadas de otra manera podrían caer en la inseguridad de la red.

Copia segura con SCP

SCP, ***Secure Copy***, es un protocolo similar a RCP, ***Remóte Copy***, cuyo objetivo es transferir archivos desde una máquina a otra, ya sean desde una máquina local a otra remota, o desde dos remotas. La diferencia entre estos dos protocolos es que SCP utiliza a SSH para crear un túnel cifrado por el que enviar los archivos, de esta manera se evita que un ***sniffer*** pueda capturar dicha transferencia.

La ejecución de SCP es realmente sencilla, su objetivo es realizar la copia de ficheros de uno o varios orígenes a una ubicación remota. Para este hecho se apoya en el protocolo SSH, que como se explicó anteriormente, proporcionando seguridad en la transferencia.

La sintaxis del comando es la siguiente ***scp [opciones] origen 1 [origen 2 ... origen N] destino.*** En los distintos orígenes se especifican las rutas, ya sean absolutas o relativas, donde se encuentran los ficheros. Los orígenes pueden ser máquinas remotas también, por lo que se debería especificar el origen de la siguiente manera ***usuario@direcciónIP:<ruta archivo remoto>.*** El destino se especifica igual que el origen, ya sea ruta local o máquina remota.

A continuación se especifican distintas opciones que pueden resultar de interés en la gestión y administración de copias seguras:

Parámetro	Descripción
-p	Preservar la fecha de modificación, acceso y modo del archivo original
-v	Modo *verbose*, el cual habilita mensajes de *debug* para diagnosticar problemas de conexión, autenticación y/o configuración
-r	Permite copiar directorios de manera remota. También copia los subdirectorios que contengan el directorio padre
-i	Selecciona el archivo desde el cual se lee una clave privada. Esta clave se le pasa a SSH para la autenticación remota
-P	Indica el puerto del servidor SSH al que se realiza la copia

Escenario 1: Copia de máquina local a remota

En el siguiente escenario se presenta la copia de una carpeta desde la máquina local en la que se está trabajando hacia una máquina remota.

```
pablo@pablo-VirtualBox:~$ scp -r /home/pablo/copiarRemoto/ pablo@192.168.0.68:/home/pablo
pablo@192.168.0.68's password:
fic1.txt                                      100%    4     0.0KB/s   00:00
fic2.txt                                      100%    4     0.0KB/s   00:00
pablo@pablo-VirtualBox:~$ pwd
/home/pablo
pablo@pablo-VirtualBox:~$ scp -r copiarRemoto/ pablo@192.168.0.68:/home/pablo
```

Fig 7.11: Envío de carpeta de local a máquina remota con SCP.

Escenario 2: Copia de máquina remota a local

En el siguiente escenario se presenta la copia de una serie de carpetas alojadas dos servidores remotos a la máquina local donde el administrador se encuentra trabajando. Cuando se especifica ***user@ip:-/carpeta,*** se está indicando con el ***home*** del usuario. En el destino se indica un V con el que se indica que la ruta donde se debe depositar el fichero descargado es la ruta actual donde se ejecuta la ***shell*** o terminal.

```
pablo@pablo-VirtualBox:~$ scp -r pablo@192.168.0.68:~/metasploit .
pablo@192.168.0.68's password:
f2.txt                                        100%    2     0.0KB/s   00:00
f1.txt                                        100%    2     0.0KB/s   00:00
f3.txt                                        100%    2     0.0KB/s   00:00
pablo@pablo-VirtualBox:~$
```

Fig 7.12: Descarga de carpetas de servidores remotos mediante SCP.

Escenario 3: Copia entre máquinas remotas

En el siguiente escenario se presenta la copia de un fichero entre dos servidores remotos, es decir, el fichero se encuentra alojado en un servidor remoto al que el administrador trabaja, y se realizará la copia sobre un servidor también remoto. En ningún caso el administrador dispone de acceso físico a

dichos servidores, pero gracias a la flexibilidad de SCP se puede realizar transferencias de archivos entre distintas máquinas remotas. Se ha utilizado autenticación con clave pública para llevar a cabo esta acción. En la imagen se puede visualizar como se leen los distintos ***banners*** de los servidores remotos a los que se accede.

```
pablo@pablo-VirtualBox:~$ scp pablo@192.168.0.69:~/secretos.txt pablo@192.168.0.68:.
Bienvenido al servicio SSH
Informatica 64
Pablo Gonzalez
```

Fig 7.13: Envío de fichero entre *hosts* remotos con SCP.

FTP seguro con SFTP

SSH File Transfer Protocolo, o protocolo de envío de ficheros mediante SSH, es un protocolo de nivel de aplicación que permite la manipulación y gestión de archivos. Estas acciones se realizan mediante una capa segura proporcionada por SSH. Con este protocolo se puede gestionar archivos con los comandos típicos de FTP en servidores remotos. Es sabido que el protocolo FTP no es seguro, ya que sus credenciales y archivos pueden ser capturados debidos al medio por el que se transmiten. Gracias a la capa SSH que se utiliza con este protocolo se asegura la información que viaja por la red.

La sintaxis de esta aplicación es la siguiente ***sftp usuario@ip.*** Como se puede visualizar es muy similar a la cadena de conexión de la aplicación ***ssh.*** Una vez conectado los comandos a utilizar son los del protocolo FTP. En la siguiente imagen se puede visualizar la conexión, con autenticación mediante clave pública, por lo que no se requiere contraseña.

```
pablo@pablo-VirtualBox:~$ sftp pablo@192.168.0.68
Connected to 192.168.0.68.
sftp> pwd
Remote working directory: /home/pablo
sftp>
```

Fig 7.14: Conexión con clave pública y SFTP.

SSHFS: El sistema de archivos de SSH

El protocolo SSH dispone de la posibilidad de gestionar el sistema de archivos de manera remota y segura. Para llevar a cabo dichas acciones se puede utilizar SSHFS ***Secure SHell File System.*** En el equipo local donde se realiza el montaje de SSHFS, la implementación hace uso del módulo de ***kernel*** FUSE. SSHFS se encuentra disponible para otras plataformas, como por ejemplo ***Mac OSX.***

La sintaxis es sencilla, tal y como se puede apreciar en la imagen, se debe ejecutar el comando ***sshfs*** seguido de la ruta remota que se quiere montar localmente y por último la ruta local que se utilizará como punto de montaje. La sintaxis es ***sshfs user@ip:<ruta remota> <ruta local o punto montaje>.***

```
pablo@pablo-VirtualBox:~$ mkdir remoto
pablo@pablo-VirtualBox:~$ sshfs root@192.168.0.61:. remoto/
The authenticity of host '192.168.0.61 (192.168.0.61)' can't be established.
RSA key fingerprint is 3e:e0:ea:5c:75:88:00:67:ba:95:6e:9b:0d:26:d9:26.
Are you sure you want to continue connecting (yes/no)? yes
root@192.168.0.61's password:
pablo@pablo-VirtualBox:~$ ls remoto/
archivos.rb           javascript.txt          mundo.rb
```

Fig 7.15: Montaje de una ruta remota mediante SSHFS.

Una vez se ha realizado el montaje, se puede utilizar cualquier tipo de operación como si la ruta remota se encontrase en local. Por ejemplo, las típicas acciones de copiado, cortado, modificación de archivos, son fácilmente realizables con este mecanismo. Los cambios que se realicen en el punto de montaje local, el cual apunta a la ruta remota, se almacenarán en la máquina remota. Todas estas acciones ocurrirán de manera transparente al usuario y mediante un túnel seguro.

Otra manera sencilla de gestionar los archivos remotos es mediante el uso del explorador de archivos. Aunque hay que recordar que los archivos se encuentran en un servidor remoto.

Montaje automático en FSTAB

Se puede montar automáticamente al iniciar la máquina o servidor una carpeta remota, para que otra aplicación u otro usuario puedan utilizar dichos recursos remotos. El fichero a configurar es / ***etc/fstab.*** Este fichero se encarga de montar ubicaciones físicas en ubicaciones lógicas en el proceso de arranque del servidor, permitiendo que las carpetas o dispositivos se encuentren disponibles automáticamente.

La edición de dicho fichero es realmente sencilla, simplemente hay que especificar la siguiente instrucción ***sshfs#username@miservidor:/ruta/remota /ruta/localfuse defaults 0 0.*** El primer campo indica el origen físico de los datos. El segundo y tercer campo indican la carpeta local donde se realizará el montaje y el tipo de sistema de archivos que se utiliza. El cuarto campo indica opciones que se utilizarán en el montaje, la ejecución del comando ***mount.*** Se ha especificado las opciones por defecto, pero se pueden utilizar opciones como ***noauto***, si no se quiere automontar, ***ro*** para solo lectura, ***noexec*** para no ejecutar, etcétera. El quinto campo indica si se puede realizar un volcado de los errores que ocurren en el sistema de archivos, dicho campo solo puede valer 0 o 1. El sexto campo indica si se debe realizar un chequeo de integridad sobre los datos del dispositivo. Si se activa con valor 1 se realizará cada x desmontajes un chequeo de la integridad de los datos.

Se necesitará utilizar autenticación mediante clave pública para que el SSHFS se monte al inicio correctamente. Por lo que de dicha forma la automatización es total en el ámbito de la administración. También hay que destacar que para que el montaje automático funcione correctamente se debe realizar con la identidad de ***root***, ya que es el único usuario que se encuentra iniciado en el arranque del sistema.

Xll forwarding con SSH

Esta opción del protocolo SSH permite al usuario conectarse a un servidor y poder levantar procesos que requieran de una interfaz gráfica para su interacción. De este modo un administrador puede ejecutar, por ejemplo, un navegador en otro servidor y la presentación de este se realizará en la máquina local, aunque el proceso se esté ejecutando en la máquina remota. Toda comunicación se realizará de manera segura, por lo que los datos de presentación del proceso viajan por la red bajo SSH.

Hay que destacar que para poder redirigir la salida de las ventanas de los procesos que se ejecutan en la máquina remota se debe habilitar la directiva ***Xll Forwarding*** en el fichero ***/etc/ssh/sshd_config.***

La sintaxis para crear una conexión SSH con una máquina remota en la que se puedan levantar procesos con GUI es la siguiente ***ssh -X usuario@ip.*** El ***flag*** X lo que indica es que se puede redirigir desde el servidor remoto hasta

Tras la conexión si se ejecuta un proceso con interfaz gráfico se recibe el PID en la terminal y se obtiene la presentación de este. El proceso se ejecuta sobre los recursos del servidor remoto, pero su interfaz gráfica se redirige a la máquina local del administrador, tal y como se puede visualizar en la imagen.

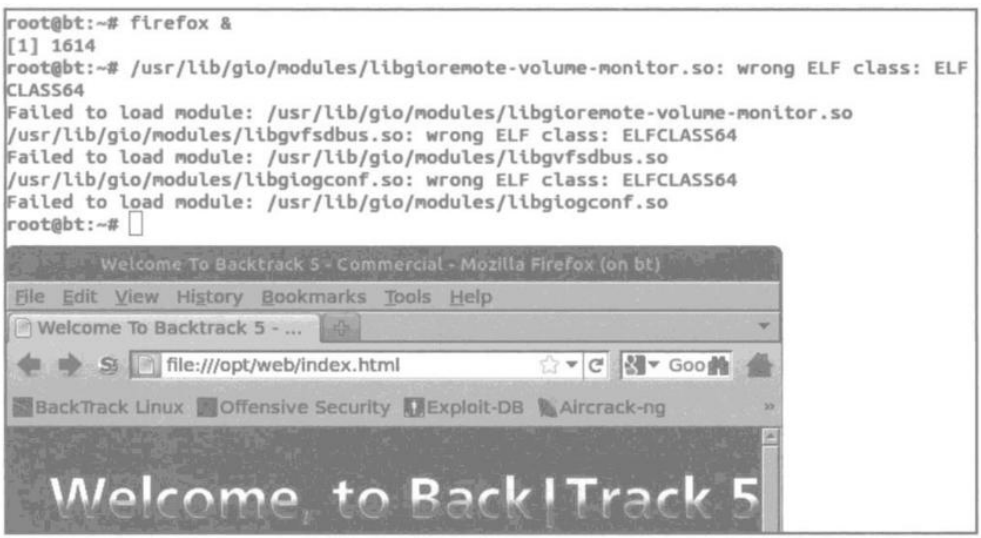


Fig 7.16: Visualización de la interfaz de un proceso remoto mediante SSH.

Fail2ban

Esta aplicación desarrollada en el lenguaje ***Python*** permite la prevención de intrusos en un sistema. Su función es penalizar la conexión, ya sea por medio de un bloqueo, de un origen que intenta realizar un proceso de fuerza bruta. En otras palabras, cuando una dirección IP o varias intentan realizar

un ataque de fuerza bruta sobre un servicio, como puede ser FTP, SSH, etcétera, esta aplicación detectará y penalizará dichos comportamientos.

Funcionamiento

Fail2ban se encarga de realizar una búsqueda en los ***logs*** de ciertas aplicaciones, las cuales se especificarán por parte del administrador en su configuración. ***Fail2ban*** contiene reglas que han sido configuradas por el usuario para aplicar penalización, la cual puede interpretarse como bloqueo de la aplicación en un determinado puerto, bloqueo para todos los puertos, etcétera. La penalización y las reglas serán definidas por el usuario.

Cuando ***Fail2ban*** detecta una serie de intentos fallidos, los cuales son predefinidos por el usuario, la aplicación determina la acción a tomar sobre la dirección IP que provocó dicha situación. Se puede, simplemente, notificar mediante un ***email*** el suceso ocurrido, denegar el acceso a la dirección IP, denegarla en determinado puerto y habilitarla en otros, mediante la utilización de ***iptables*** o el ***firewall*** que corresponda. Además, se puede configurar las prohibiciones pasado un determinado período. Esta acción es interesante por si fuera un falso positivo. Todo es configurable mediante ***Scripts*** de ***python.***

Directivas de Fail2ban

La instalación de ***Fail2ban*** se realiza a través de los repositorios de las distribuciones de ***GNU/ Linux***, por lo que con la ejecución del comando ***apt-get installfaiUban*** es suficiente para instalar el servicio.

En este apartado se reflejan las directivas más interesantes de ***Fail2ban*** para configurar las reglas necesarias para prohibir el acceso a máquinas que intentan realizar fuerza bruta sobre el servidor SSH.

Las reglas se configuran en el archivo ***/etc/fail2ban/jail.conf.*** Su distribución es sencilla, el fichero se divide en áreas. Para las directivas correspondientes con el protocolo SSH se utiliza el identificador "[SSH]" para señalar en que área se encontrará configurado lo relacionado con dicho protocolo.

```
[ssh]

enabled  = true
port     = ssh
filter   = sshd
logpath  = /var/log/auth.log
maxretry = 3

fwban = iptables -I INPUT -s <ip> -j DROP

fwunban = iptables -D INPUT -s <ip> -j DROP
```

Fig 7.17: Directivas configuradas de *Fail2ban* para SSH.

La directiva ***port*** indica el puerto por el que el servidor está escuchando, la cual por defecto presenta el puerto 22, pero puede ser cambiado si el servidor funciona en otro puerto. La directiva ***logpath*** indica la ruta donde se encuentra el fichero de ***log*** donde ***Faittban*** buscará los intentos de conexión masivos. La directiva ***maxretry*** indica el número de intentos que puede realizar una dirección IP antes de ser baneada.

La directiva ***fwban*** indica la acción a realizar cuando se detecte un número, definido por ***maxretry***, de intentos elevado. Un ejemplo sería indicar que la dirección no puede conectar con la máquina servidora, o por ejemplo no puede conectar solo con dicho servicio.

La directiva ***fwunban*** indica la acción a realizar cuando se sobrepase el tiempo de castigo o tiempo de baneo. Generalmente, se eliminar las reglas activadas en el momento de realizar el baneo sobre el cliente.

La directiva ***bantime*** indica el número de segundos que la dirección IP es castigada sin conectividad con el servidor. La directiva ***ignoreip*** indica una dirección IP o una red que es excluida del análisis realizado por ***Fail2ban.***

Existe la posibilidad en este servicio de notificar mediante correo electrónico cuando el servicio queda inhabilitado o parado.

Escenario con Fail2ban

En el siguiente escenario se presenta un servidor SSH implementado sobre un ***Ubuntu*** y una máquina ***Backtrack*** que se utiliza de cliente para las conexiones SSH. ***Ubuntu*** ha sido configurado para que al tercer intento de contraseña fallido se prohíba la conexión de la máquina ***Backtrack*** con el servidor.

```
pablo@192.168.1.36's password:
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
pablo@192.168.1.36's password:
debug1: Authentications that can continue: publickey,password
Permission denied, please try again.
pablo@192.168.1.36's password:

^C
root@bt:~# ssh -v pablo@192.168.1.36
OpenSSH_5.3p1 Debian-3ubuntu6, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 192.168.1.36 [192.168.1.36] port 22.
^C
root@bt:~#
```

Fig 7.18: Máquina baneada con *Fail2ban*.

En la siguiente imagen se puede visualizar como queda registrada una dirección IP y la fecha en la que ha sido baneada. Este hecho es importante ya que cuantos más datos queden almacenados más sencillo será llevar a cabo un proceso forense sobre lo que ha sucedido.

```
2013-02-09 17:41:26,534 fail2ban.jail   : INFO   Creating new jail 'ssh'
2013-02-09 17:41:26,535 fail2ban.jail   : INFO   Jail 'ssh' uses Gamin
2013-02-09 17:41:26,545 fail2ban.filter : INFO   Added logfile = /var/log/auth.log
2013-02-09 17:41:26,545 fail2ban.filter : INFO   Set maxRetry = 3
2013-02-09 17:41:26,546 fail2ban.filter : INFO   Set findtime = 600
2013-02-09 17:41:26,546 fail2ban.actions: INFO   Set banTime = 600
2013-02-09 17:41:26,569 fail2ban.jail   : INFO   Jail 'ssh' started
2013-02-09 17:41:54,626 fail2ban.actions: WARNING [ssh] Ban 192.168.1.41
2013-02-09 17:51:55,516 fail2ban.actions: WARNING [ssh] Unban 192.168.1.41
2013-02-09 17:55:55,850 fail2ban.actions: WARNING [ssh] Ban 192.168.1.41
pablo@pablo-VirtualBox:/etc/fail2ban$
```

Fig 7.19: Log de *Fail2ban*.

4. Tunneling

El ***tunneling*** es el proceso por el que se utiliza el protocolo SSH para fortificar la comunicación entre otros protocolos inseguros. Por ejemplo, se puede crear un túnel de tipo SSH y mediante el uso de dicho túnel realizar peticiones HTTP, SMTP, FTP, etcétera. Es una manera de asegurar la comunicación de manera sencilla como se podrá comprobar en las siguientes líneas.

SSH: tunneling

Los túneles pueden ser locales o remotos. En los túneles locales, la máquina del administrador o cliente redirecciona un puerto local hacia un puerto de la máquina remota, a la que se disponga de acceso. En los túneles remotos se redirecciona un puerto remoto de la máquina remota, a la que se disponga de acceso, a un puerto de la máquina local, dónde se encuentra el administrador.

Túnel local

La sintaxis para la creación de un túnel local hacia el puerto remoto de la máquina remota es la siguiente ***ssh -L [dirección es cucha:]puerto es cucha .máquina remota:puerto remoto usuario@ip.***

El escenario ideal para la utilización de túneles SSH es el siguiente:

- Máquina A con conexión a Internet se quiere conectar a la máquina C, la cual se encuentra detrás de un servidor que impide el paso. En otras palabras, la máquina C no es accesible desde Internet.
- El usuario de la máquina A dispone de cuenta en una máquina B. La máquina B es el servidor que proporciona la salida a Internet a la máquina C. La máquina A creará un túnel con la máquina B y a través del ***forwarding*** de la máquina B se llegará a la máquina C.
- Lo importante es proteger la comunicación en Internet entre la máquina Ay B. La comunicación entre B y C puede ir sin cifrar, ya que se encuentra en una Intranet. Aunque también puede ser altamente recomendable fortificar dicha comunicación.

- La sintaxis, por ejemplo, sería ***ssh -L 8080:MáquinaC:80 user@máquinaB.*** De esta forma se creará un túnel desde la máquina A hasta C, utilizando la máquina B para dar acceso a la red interna.
- En el ejemplo se utiliza el puerto local 8080 y el puerto 80 en la máquina destino o final del túnel. Se quiere acceder a un servidor web interno de la máquina C, por lo que en la máquina A se abre el navegador y utilizando el recurso http://localhost:8080**,** se obtiene el sitio web que dispone la máquina C en el puerto 80.

Para ejemplificar este hecho más se realizará paso a paso un ejemplo que simule lo especificado anteriormente. En primer lugar se debe crear el túnel, pudiéndose crear más de uno, simplemente añadiendo instrucciones del tipo ***-L puerto escucha: dirección remota: puerto remoto.*** En la imagen se puede visualizar como el final del túnel es el equipo con el que se quiere realizar la conexión, en este caso se quiere proteger la comunicación HTTP. Por último se indica a través de que máquina se realizar el ***port-forwarding***, en la que se debe de disponer de un usuario.

```
root@pablo-VirtualBox:/home/pablo# ssh -L 8080:192.168.0.58:80 192.168.0.60
root@192.168.0.60's password:
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:34:20 EST 2012 x86_64 GNU/Linux

  System information as of Tue Mar 19 06:51:16 EDT 2013

  System load:    0.0       Processes:           104
  Usage of /home: unknown   Users logged in:     1
  Memory usage:   33%       IP address for eth0: 192.168.0.60
  Swap usage:     0%

  Graph this data and manage this system at https://landscape.canonical.com/
Last login: Tue Mar 19 06:51:10 2013 from 192.168.0.58
root@bt:~#
```

Fig 7.20: Creación túnel local hacia el destino.

Una vez creada la sesión interactiva con el equipo que se encuentra en el medio, el que se utilizará para realizar el ***forwarding*** mediante el túnel SSH, se puede visualizar como se ha habilitado el puerto 8080, en este caso de manera local, por lo que se si envía algo a ***localhost*** al puerto 8080, esto será reenviado por el túnel hasta el destino.

```
pablo@pablo-VirtualBox:~$ netstat -a | grep ssh
tcp        0      0 *:ssh                   *:*                     ESCUCHAR
tcp        0      0 pablo-VirtualBox.:59353 192.168.0.60:ssh        ESTABLECIDO
tcp6       0      0 [::]:ssh                [::]:*                  ESCUCHAR
unix  2      [ ACC ]     FLUJO      ESCUCHANDO     9755     /tmp/keyring-KSCsNx/ssh
unix  2      [ ACC ]     FLUJO      ESCUCHANDO     9377     /tmp/ssh-cbEwABcv1331/agent.1
331
pablo@pablo-VirtualBox:~$
```

Fig 7.21: Conexión local y socket.

Por último para visualizar que es correcto todo lo mencionado anteriormente se prueba a utilizar el navegador introduciendo como dirección URL ***localhost:8080.*** Se obtiene la página de prueba de ***Apache***, por lo que la conexión HTTP está siendo protegida por este túnel (entre las máquinas A y

B). Este hecho se puede visualizar al abrir un ***wireshark*** en el nodo intermedio de la conexión, no se vería el tráfico HTTP entre la máquina A y C. Se vería tráfico cifrado por SSH entre la máquina A y B, y como la máquina B envía a la máquina C el tráfico sin SSH, ya fuera del túnel.

Fig 7.22: Obtención de recurso web a través de un túnel SSH.

Otra opción interesante es permitir que desde el equipo local se conecten otros equipos al remoto (o máquina C en el ejemplo anterior). Para este hecho simplemente se debe realizar el túnel de la siguiente manera ***ssh -L *:8080:maquinaC:80 user@maquinaB.*** Por lo que ahora cuando alguien contacte al ***socket*** local mediante http://nombreDominioMaquinaA:8080 será reenviado a la máquina C.

Túnel remoto

El túnel remoto generará un ***socket*** a la escucha en un servidor remoto, el cual al recibir una petición reenviará el tráfico mediante SSH a la máquina local donde se encuentre el administrador. La sintaxis es sencilla ***ssh -R puerto local:máquina remota:puerto remoto***.

Un ejemplo sería querer que una máquina en Internet llegase al interior de la intranet a través de una máquina puente de la propia red nuestra con una interfaz hacia Internet. El escenario es el siguiente:

- Máquina A, la máquina se encuentra en una red interna.
- Máquina B, máquina con conectividad con A e interfaz en Internet.
- Máquina C, máquina externa con conectividad con B, pero no con A.

El administrador que dispone de una cuenta en la máquina B para realizar la siguiente operación ***ssh -R 8080:maquinaA:80 user@maquinaB***. Cuando en la máquina B, cualquier proceso local, se ejecute una petición al puerto 8080 se redirigirá por el túnel hasta la máquina A y esta realizará la petición al puerto 80 de la máquina A. Hay que recordar que si se quiere utilizar la máquina B como puente para que otras máquinas puedan usar dicho túnel se debería ejecutar la instrucción ***ssh -R *:8080:maquinaA:80 user@maquinaB.***

Como se puede comprobar la diferencia entre un túnel remoto y un túnel local es simplemente dónde se genera el ***socket*** ya que el funcionamiento es exactamente el mismo. Hay que recordar que hay que habilitar la directiva ***GatewayPorts,*** que por defecto viene a no y modificarla por ***yes.*** Si esta directiva no se encuentra habilitada en el fichero ***/etc/ssh/sshd_config*** el servidor SSH no funcionará para el reenvío en esta modalidad.

Túneles TCP/IP con port forwarding mediante SSH

En este apartado se van a comentar algunas de las configuraciones más interesantes para el uso de los túneles SSH en el entorno laboral. Los escenarios más usuales para utilizar los túneles SSH son los siguientes:

- Usar un túnel externo para atravesar el ***firewall*** y acceder a un recurso que ofrece un equipo de la intranet, ***ssh -L 7900:localhost:5900 usuario@interno 1.***
- Usar un túnel externo para atravesar el ***firewall*** y acceder a un recurso de una intranet. Pero además, cualquier equipo puede conectarse a dicho equipo de la intranet a través de una conexión con el equipo externo, ***ssh -L 0.0.0.0:7900:localhost:5900 usuario@interno 1.***
- Usar un túnel interno para que equipos de fuera de la red se conecten al interior. Esta acción se realizará mediante túneles remotos, explicados anteriormente, ***ssh -R 7900:localhost:5900 usuario@internol.***
- Túnel interno para que equipos de fuera de la red se conecten al interior, pero además que otros equipos del exterior puedan conectarse a través del externo remoto, ***ssh -R 0.0.0.0:7900:localhost:5900 usuario@interno 1.***
- Utilizar un túnel para que desde una máquina externa se pueda atravesar el ***firewall*** y realizar el ***forwarding*** en una máquina interna y que esta reenvíe el tráfico a una segunda máquina interna, ***ssh -L 7900:interno2:5900 usuario@internol.***

5. SOCKS con SSH

A continuación se muestra una técnica similar a la anterior, pero puede que en determinadas circunstancias resulte más flexible en su utilización. Se trata de una de las funcionalidades más utilizadas y consiste hacer pasar el tráfico a través de un servidor ***SSH.*** Una aplicación muy popular de este método es su utilización para salir a Internet bajo entornos inseguros o bien bloqueados por ***firewalls, proxies***, etc.

Cuando se utiliza ***SSH*** de este modo, la máquina cliente ***SSH*** actuará al mismo tiempo como ***proxy SOCKS***, creando un ***socket*** de escucha localmente. ***SOCKS*** es un protocolo posicionado entre la capa de transporte y aplicación que permite redirigir y encapsular tráfico de red. Su utilización, frente a alguno de los métodos de ***tunneling*** ya mostrados, reside en que en este caso las redirecciones de tráfico son dinámicas para todas aquellas aplicaciones que, en principio, implementen la funcionalidad de cliente ***SOCKS.*** Es decir, no será necesario crear túneles individuales para cada una de las aplicaciones o conexiones a realizar.

Por lo tanto, se puede concluir que los elementos necesarios para utilizar ***SOCKS*** sobre ***SSH*** son:

- Servidor ***SSH.*** Hará las veces de ***router*** para encaminar el tráfico que llegue desde el cliente ***SSH.*** Todo ello mediante el uso interno del protocolo ***SOCKS.***

- Cliente ***SSH.*** Al conectar con el servidor utilizando el parámetro ***-D*** en el comando ***ssh***, se creará un ***socket*** en la máquina local. Dicho ***socket*** será el encargado de recibir las peticiones de los clientes ***SOCKS*** y al mismo tiempo enviar dichas peticiones encapsuladas bajo el protocolo ***SOCKS*** a través de la conexión ***SSH.*** Por supuesto las respuestas a dichas peticiones serán gestionadas a la inversa.
- Aplicación. Como se comentaba anteriormente, las aplicaciones con las que se desee utilizar ***SOCKS*** deberán implementar el protocolo para conectar con el servidor proxy generado localmente. Un ejemplo de aplicación muy común es un navegador web, por ejemplo Firefox.

Un escenario de utilización de ***SSH*** y ***SOCKS*** para navegación web podría ser como el planteado en la siguiente imagen, en la que el equipo portátil conectará con www.informatica64.com a través del servidor ***SSH.***

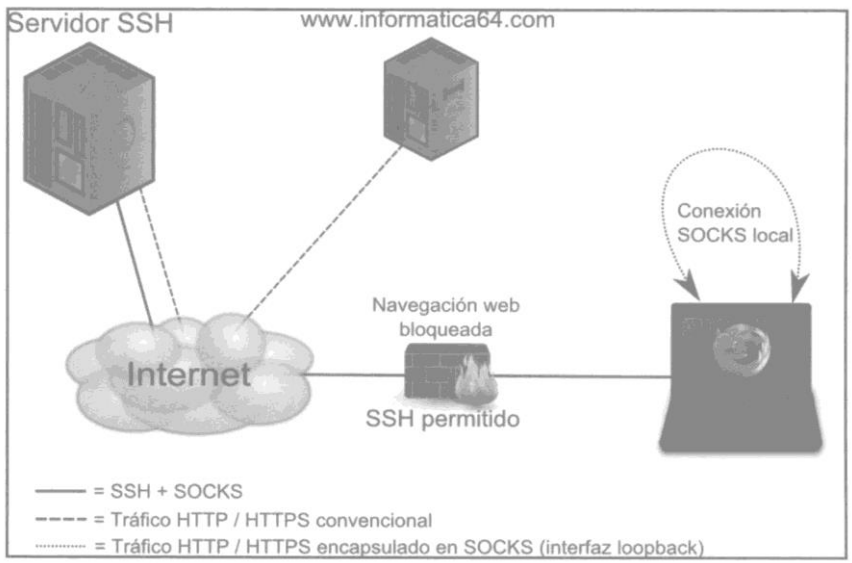


Fig 7.23: Navegación web a través de servidor SSH con SOCKS.

Como nota adicional, cabe destacar que se utiliza ***SOCKS5***, ya que mejora a la versión anterior añadiendo soporte para el protocolo ***UDP*** y autenticación entre otros. Aunque realmente el funcionamiento de ***SOCKS*** es totalmente transparente cuando se combina con ***SSH.***

Habilitando y utilizando SOCKS

Con la configuración por defecto del servidor ***OpenSSH*** es posible utilizar el protocolo ***SOCKS*** cuando el cliente ***SSH*** así lo indique. Por lo tanto, en este punto solo se atenderá a la conexión que establezca el cliente.

Creando el túnel con soporte para SOCKS

Para que el cliente empiece a utilizar la funcionalidad debe establecer la conexión con el servidor ***SSH*** utilizando el parámetro ***-D*** en el comando ***ssh.*** A dicho parámetro le seguirá un número de puerto.

```
$ ssh carlos@192.168.87.55 -D 1080
```

Si el usuario ***carlos*** se ha autenticado de forma correcta con el servidor ***SSH 192.168.87.55***, se habrá establecido la conexión ***SSH*** y adicionalmente estará disponible un nuevo ***socket***, utilizando el puerto ***1080/tcp***, en la máquina cliente.

En efecto, se trata del proxy ***SOCKS*** al que se deberán enviar las conexiones a encaminar por el túnel ***SSH.*** Es posible optimizar el establecimiento de este túnel usando parámetros para indicar que no se trata de una sesión interactiva pues solo es un túnel, utilizar ***shells*** con permisos limitados, usar compresión, etc.
Por otra parte, podrá comprobarse que el proxy está disponible mediante la siguiente línea.

```
$ netstat -putna | grep 1080
tcp    0      0      127.0.0.1:1080          0.0.0.0:*       LISTEN  10740/ssh
tcp6   0      0      ::1:1080                :::*            LISTEN  10740/ssh
```

Es decir, el ***proxy*** está escuchando en local. Puede obtenerse más ayuda sobre el parámetro ***-D*** en el ***man*** de ***ssh.***

Configuración de la aplicación para utilizar proxy SOCKS

Dependiendo de cada aplicación, la configuración de la misma para utilizar un ***proxy SOCKS*** será diferente en cada caso. De cualquier modo, de forma general habría que indicar que el ***proxy SOCKS*** es ***localhost*** y el puerto es el ***1080/tcp.***

Se debe tener en cuenta que en el caso de los navegadores, la consultas DNS no se envían a través del túnel. Como se comentó anteriormente, ***SOCKS5*** permite encapsular UDP y por lo tanto se podría configurar Firefox en este caso para que dichas consultas DNS desde el navegador se encaminen a través del servidor ***SSH.***

Por otra parte, desde entornos de escritorio como ***Gnome*** es posible configurar de forma global un ***proxy SOCKS***, así aquellas aplicaciones que estén configuradas de forma automática podrán utilizarlo siempre que lo soporten.

Un ejemplo de esta configuración podría ser el caso de ***Firefox***, cuya pantalla de configuración tendría un aspecto como el siguiente al configurarlo para usar un ***proxy SOCKS.***

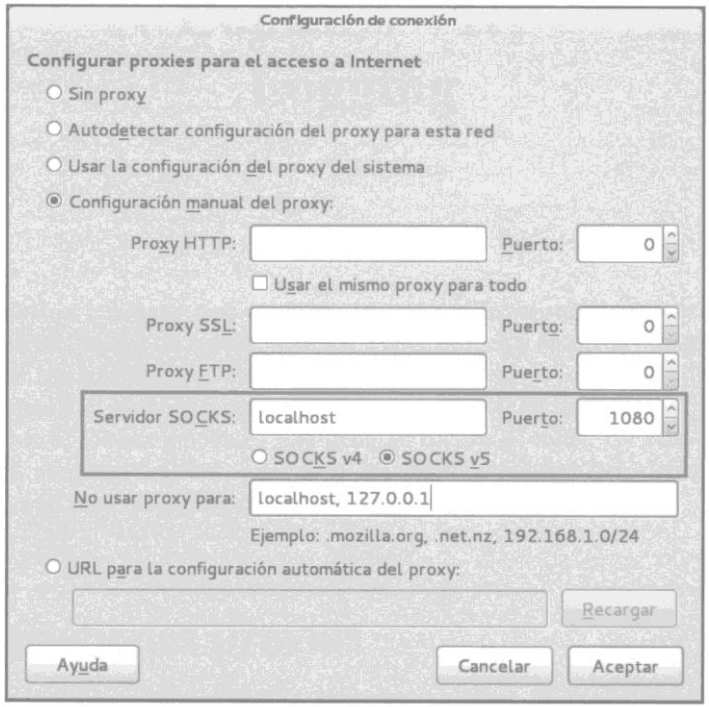


Fig 7.24: Configuración de Firefox para SOCKS.

Aplicaciones sin soporte para SOCKS

Uno de los inconvenientes que existen al utilizar un ***proxy SOCKS*** surge cuando determinadas aplicaciones no disponen de soporte para ello. En estos casos existen aplicaciones que modificarán llamadas necesarias del sistema, para que una aplicación sin soporte para ***SOCKS*** pueda utilizarlo.

Algunas de estas aplicaciones son ***tsocks***, el cliente ***dante*** o ***proxychains.*** Una vez instalada y configurada la utilidad ***tsocks*** por ejemplo, se utilizaría del siguiente modo.

```
$ tsocks telnet informatica64.com
```

Es decir, se estaría estableciendo una conexión mediante ***telnet***, sin soporte nativo para ***SOCKS***, a través del túnel ***SSH.*** Para ello, como se comentaba previamente se habrá indicado qué ***proxy SOCKS*** utilizar en el fichero ***/etc/tsocks.conf.***

En los casos anteriores, podría verificarse que ***SOCKS*** está actuando verificando capturas de red en ***Wireshark.*** Podría comprobarse capturando la interfaz de ***loopback*** que efectivamente existe tráfico de tipo ***SOCKS.*** Por otro lado, al realizar una captura de la interfaz física del equipo se observaría una elevada cantidad de datagramas de tipo ***SSH***, que indicaría que el tráfico se está enviando a través del túnel.

Capítulo VIII
Logging

A la hora de administrar cualquier tipo de sistema resulta indispensable conocer en todo momento la actividad del mismo. Inicios de sesión, accesos a determinados recursos e intentos de ataque por ejemplo, darán a un administrador de sistemas una visión global de lo que sucede en un entorno. A ese registro de eventos o sucesos se le conoce como ***logging.*** Son los diferentes servicios y utilidades del sistema entre otros, los componentes que generan dichos eventos. En este último capítulo se proporcionarán las nociones básicas, así como las configuraciones del sistema que permitirán disponer de un entorno para el ***logging*** de eventos adecuado.

1. Consideraciones previas

Existen una serie de aspectos que deberían considerarse antes de configurar un sistema para la gestión de ***logs*** o eventos.

- Almacenamiento independiente. Los ***logs*** se almacenan de forma predeterminada en ficheros de texto plano. Su tamaño y crecimiento dependerá del número de aplicaciones que generan ***logs***, carga de la máquina y los servicios ofrecidos, además de la información que se esté registrando. Es recomendable por lo tanto aislar el sistema de ficheros en el que se almacenarán los ***logs*** para evitar colapsos en particiones como la raíz.
- Rotación de ***logs.*** Por defecto los ***logs***, cuando son almacenados en ficheros de texto plano, se almacenan de forma secuencial. Es decir, el fichero de log irá creciendo indefinidamente. Ese comportamiento por defecto conllevaría la generación de ficheros de gran capacidad, pérdida de rendimiento, además de pérdida de eficiencia en caso de necesidad de realizar búsquedas.
- Métodos de ***logging*** de aplicaciones. Es necesario tener en cuenta cómo ciertas aplicaciones registran sus eventos en el sistema. Como se verá posteriormente existen diferentes ficheros de log estándar donde se almacenará cierta información sobre diferentes tipos de eventos disparados por una aplicación o mecanismo del sistema. Dependiendo del tipo de aplicación es posible que la información se reparta de forma íntegra en esos ficheros utilizando un mecanismo de log local como ***rsyslog***, que se utilicen mecanismos y ficheros propios de la aplicación para registrar eventos, o bien que sea una combinación de ambos. Es necesario

atender a este tipo de excepciones sobre todo a la hora de rotar ***logs*** o implementar un sistema centralizado de ***logs.***

- Alertas. Es posible realizar análisis en los ***logs*** de forma periódica en busca de ciertos patrones, siendo posible el envío de correos ante determinadas situaciones. Existe la posibilidad de implementar esta funcionalidad con un nuevos mecanismos, o bien utilizar la infraestructura ***HIDS*** implementada en capítulos anteriores.
- Infraestructura de ***logs.*** Dependiendo del número de máquinas a analizar mediante ***logs*** es posible que sea prácticamente imposible revisarlos en cada una de ellas. Es por ello que existe la posibilidad de centralización de ***logs***, almacenamiento mediante otros mecanismos y no en ficheros de texto, análisis mediante aplicaciones web, etc.

2. rsyslogd

Se trata de un servicio encargado del manejo de ***logs*** del sistema. Su función consiste en organizar los eventos generados por determinados servicios o aplicaciones en diferentes contenedores. Los contenedores podrán ser ficheros de texto o bases de datos. La organización por su parte se realizará atendiendo a una serie de parámetros indicados por la propia aplicación que lance el evento, ***rsyslogd*** recogerá entonces esa información y dependiendo de su configuración almacenará dicho evento en ficheros, etc.

Por supuesto, ***rsyslogd*** se encargará de la gestión de ***logs*** para aquellas aplicaciones o servicios que utilicen este método, ya que como se comentaba anteriormente existen aplicaciones que gestionan sus propios ficheros de log utilizando sus propios métodos.

Existen varias alternativas para la gestión de ***logs*** en un sistema ***Linux***, sin embargo se elige ***rsyslog*** por su estandarización en las distribuciones más importantes. Entre ellas Debían y Red Hat.

Clasificación de mensajes. Facility y severity.

Para la organización de ***logs***, ***rsyslog*** recibirá mensajes por parte de las aplicaciones que utilicen la API ***syslog.*** Para ello, como ya se ha comentado, dichas aplicaciones especificarán una serie de parámetros en los eventos que generen, con la finalidad de que ***syslog*** pueda clasificarlos mediante las reglas que se configuren.

Facility debe entenderse como una clasificación para los mensajes. Existen una serie de ***facilities*** predeterminadas como ***cron***, ***kern***, ***daemon***, ***mail***, ***authpriv***, ***auth***, etc. que serán utilizadas por las aplicaciones dependiendo del tipo de mensaje que se genere. Además de esas ***facilities*** predeterminadas existen otras que podrán ser utilizadas libremente por el administrador de sistemas. Son las ***facilities localN***, siendo ***N*** números del ***0*** al 7.

Severity indica el nivel de gravedad o severidad de un evento. Este indicativo ayuda a diferenciar entre mensajes informativos de mensajes de emergencia por ejemplo. Algunas de las ***severities*** existentes son ***debug***, ***info***, ***warn***, ***err*** o ***emerg.***

Utilizando combinaciones de ***facility*** y ***severity*** podrán clasificarse los eventos que reciba ***rsyslogd.*** Para entender mejor la finalidad de ambos conceptos, la siguiente podría ser una línea de configuración de ***rsyslog.***

```
local0.info     /var/log/ejemplo-local0.log
```

Con esa línea se indica que todos los eventos con ***facility localO*** y ***severity info*** sean almacenados de forma secuencial en el fichero ***/var/log/ejemplo-local0.log.***

Es posible que este tipo de clasificación parezca algo rígida en cuanto a configuración se refiere, por lo que desde hace ya algunas versiones ***rsyslog*** ofrece la oportunidad de clasificar o filtrar por propiedades en los mensajes de log como el ***hotsname*** o el nombre de la aplicación. Todo ello utilizando expresiones lógicas, expresiones regulares, etc. Por defecto no es el mecanismo utilizado en la mayoría de distribuciones por el hecho de mantener la compatibilidad con sistemas antiguos. Es muy recomendable leer la documentación de ***rsyslog***, así como el ***man 5 rsyslog*** para la utilización de este nuevo método.

Configuración de rsyslogd

Para configurar el comportamiento de ***rsyslog*** se utiliza principalmente el fichero ***/etc/rsyslog.conf*** En el caso de Debian de forma adicional pueden utilizarse los ficheros contenidos en ***/etc/rsyslog.d*** para crear configuraciones fragmentadas o por servicios.

El fichero principal de configuración se divide en cuatros partes, no todas obligatorias.

- Directivas globales. Se define el comportamiento del demonio ***rsyslog.*** Es donde se cargan los módulos para agregar características, se establecen los permisos de los ficheros de log creados, etc.
- Plantillas o ***templates.*** Permiten modificar la salida de los mensajes de log atendiendo a diferentes parámetros como son el nombre de programa, ***host***, etc. También es posible utilizarlas con la finalidad de crear ficheros de log dinámicos por cada máquina o almacenamiento en bases de datos entre otros.
- Canales de salida, ***output channels.*** Definen canales que podrán ser utilizados posteriormente en los filtros o clasificación de los mensajes en lugar de utilizar un nombre de fichero por ejemplo. Es útil pues pueden definirse opciones como el tamaño máximo que puede tener un fichero de log y qué hacer cuando este se supera mediante un script.
- Reglas. Se indica la acción a realizar cuando se hace coincidir una ***facility*** y ***severity*** por ejemplo. Una regla ejemplo podría ser la mostrada anteriormente utilizando ***localO.info.***

Las acciones pueden ser el almacenamiento en ficheros, en una máquina remota con rol de servidor ***rsyslog***, ***templates*** o plantillas definidas anteriormente, etc.

Algunas de las líneas contenidas por defecto en el fichero de configuración son las siguientes.

```
$FileOwner root
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
```

Se tratan de ciertas opciones globales en las que se definen las características de los ficheros y directorios creados de forma automática por ***rsyslog.***

Las siguientes líneas en cambio corresponden a reglas, las mencionadas combinaciones de filtros y acciones.

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
daemon.*                        -/var/log/daemon.log
```

1. En la primera línea se indican que los eventos relacionados con la autenticación del sistema y se almacenen en el fichero ***/var/log/auth.log.*** Es decir, son los eventos que generan las aplicaciones con ***facility auth*** o ***authpriv*** y todas las ***severities*** de ambas ***facilities.***
2. En esta línea se introduce el símbolo que delimita conjuntos de filtros compuestos por <***facility***>.<***severity***>. El símbolo " " por su parte permite agrupar varias ***facilities*** para indicar una sola ***severity*** o el carácter comodín en su lugar, *. Por su parte el símbolo indica que no se utilice la sincronización de forma inmediata para la escritura en el disco. Por defecto no surte efecto, pues está desactivado con la opción global ***SActionFileEnableSyn.*** En definitiva esta línea escribirá todos los eventos salvo los correspondientes a autenticación en el fichero ***/var/log/ syslog.*** Se descartan las ***facilities auth*** y ***authpriv*** al indicar como ***severity none.***
3. Es una línea sencilla. Todos los eventos de la ***facility daemon*** serán escritos en el fichero ***/var/log/daemon.log.***

La siguiente entrada conseguirá que un evento con ***severity emerg*** aparezca en todas las terminales interactivas. De igual modo aparecerá en el fichero ***/var/log/syslog*** debido a la segunda línea del ejemplo anterior.

```
*.emerg                         *
```

Un caso de ***severity emerg*** sucede cuando se reinicia o apaga la máquina. En todas las terminales operativas en ese momento aparecerá un mensaje indicando el reinicio o apagado.

En este instante los ***logs*** se almacenarán en diferentes ficheros dependiendo de la configuración. Dicho almacenamiento se realizará de forma secuencial y en principio no existirá un límite para el crecimiento de los ficheros. Sería necesario por lo tanto algún mecanismo con el que rotar los ficheros de ***log.***

3. Rotación de logs

Existe una herramienta muy conocida llamada ***logrotate*** que está instalada por defecto en muchas distribuciones entre las que se incluye Debian. En este punto se explicará el funcionamiento de ***logrotate*** mediante algunos de los ficheros de configuración por defecto, que conllevarán a una rotación de los ***logs*** del sistema desde el momento en el que este se instala.

La configuración de ***logrotate*** está compuesta por el fichero principal ***/etc/logrotate.confy*** los ficheros contenidos dentro del directorio ***/etc/logrotate.d/.*** En dichos ficheros se indican características como el intervalo entre la rotación de los ***logs***, el tamaño máximo que deben ocupar, el número de ***logs*** almacenados tras cada rotación, etc.

Una vez establecida una configuración utilizando los ficheros de ***logrotate***, será necesario configurar una tarea automática para ejecutar ***logrotate*** y la configuración establecida. Por defecto se utiliza el fichero ***/etc/cron.daily/logrotate*** que contiene las siguientes líneas.

```
#!/bin/sh

test -x /usr/sbin/logrotate || exit 0
/usr/sbin/logrotate /etc/logrotate.conf
```

Es decir, de forma diaria se ejecutará el script mostrado en el que se ejecuta la aplicación ***logrotate*** utilizando la configuración contenida en ***/etc/logrotate. conf***

Ficheros de configuración de logrotate

Existe una configuración global que afectará a todos los ficheros contenidos bajo el directorio ***/var/ log***, sin embargo la mayoría de las aplicaciones que escriben sus propios ficheros de log o utilizan ***rsyslog*** agregan configuraciones propias para la rotación de ***logs.***

/etc/logrotate.conf

En este fichero se define una configuración global, siendo su contenido el siguiente.

```
...
weekly
```

```
rotate 4
create

include /etc/logrotate.d

/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}
```

Los tres primeros parámetros corresponden a la configuración general para todos los ficheros contenidos en ***/var/log.***

- ***weekly*** indica que los ficheros se rotarán semanalmente. Esto hará que cada semana el fichero de log actual sea renombrado. Por ejemplo ***/var/log/syslog*** será renombrado a ***Ivar/log/syslog.l.*** Si ***/var/log/syslog. 1*** ya existía, este será renombrado como ***/var/log/syslog. 2*** y así sucesivamente.
- ***roíate 4*** indica que como máximo existirán cuatro ficheros de ***log*** rotados. Continuando con el ejemplo de ***/var/log/syslog***, podrá existir como máximo el fichero ***/var/log/syslog. 4.***
- ***create*** indica que al renombrar el fichero de log se creará uno nuevo para que ***rsyslog*** continúe trabajando con normalidad.

Por su parte, la línea ***inelude /etc/logrotate.d*** permite leer la configuración de los ficheros contenidos en el directorio indicado. Dentro de él se encontrarán configuraciones segmentadas y propias de algunas aplicaciones o servicios, como por ejemplo ocurre con el servidor web ***Apache.***

Los dos últimos bloques establecen las opciones para rotar los ficheros binarios de ***log /var/log/ wtmp*** y ***/var/log/btmp.*** En ambos casos se rotarán mensualmente, almacenando solo una rotación e ignorando en caso que el fichero de log no exista con la directiva ***missingok.*** Adicionalmente se crearán los ficheros con los permisos indicados mediante ***create.***

/etc/logrotate.d/apache2

Se trata de la configuración de ***logrotate*** para los ficheros de ***log*** generados por ***Apache.*** Este fichero de configuración es copiado por defecto durante la instalación de ***Apache.*** En concreto mediante el paquete ***apache2.2-common.*** Su contenido es el siguiente.

```
/var/log/apache2/*.log {
 weekly
 missingok
 rotate 52
 compress
 delaycompress
 notifempty
 create 640 root adm
 sharedscripts
 postrotate
        /etc/init.d/apache2 reload > /dev/null
 endscript
}
```

Las tres primeras directivas además de ***create*** han sido explicadas, tan solo varía el número de rotaciones que se almacenarán. Se almacenarán ficheros de log de ***Apache*** de las últimas 52 semanas. El resto de directivas poseen el siguiente significado.

- ***compress*** y ***delaycompress.*** Indican que los ficheros rotados serán comprimidos con ***gzip.*** Si como en este caso se especifica además ***delaycompress***, la compresión del fichero actualmente rotado no se realizará hasta la siguiente rotación. Los ficheros que sean comprimidos poseerán una nomenclatura como ***/var/log/apache2/access.log.2.gz.***
- ***notifempty.*** Especifica que el fichero no será rotado si este no contiene información.
- ***sharedscripts,postrotate.*** Cuando ***logrotate*** realiza una rotación de log es posible ejecutar una o varias acciones. En este caso se recargará la configuración de ***Apache*** cuando se roten todos los ficheros de ***log.*** La entrada ***postrotate*** indica la ejecución después de la rotación y por su parte ***sharedscripts*** especifica que dicha acción será realizada una sola vez cuando se roten todos los ficheros de ***log*** indicados al inicio del bloque.

Existe información detallada acerca de las directivas en ***man logrotate.*** El administrador podrá crear sus propias configuraciones o bien modificar las ya existentes. Cuando se trata del almacenamiento de ***logs*** se debe tener en cuenta el periodo de almacenamiento de los mismos según dictan ciertas leyes de protección de datos en caso de manejar datos de usuario, accesos públicos, etcétera.

Output channels y logrotate

Uno de los problemas que presenta ***logrotate*** reside en que su ejecución es a modo de tarea programada y no en tiempo real. Esto puede suponer un peligro en caso de que un determinado fichero de ***log*** crezca de forma descontrolada debido a un error en un servicio, algún tipo de ataque, etc. Para estas situaciones es posible utilizar el ya mencionado ***Monit***, pudiendo controlar el crecimiento de un fichero mediante su configuración. Sin embargo, aquí se explica cómo podría conseguirse una rotación de determinados ficheros de ***log*** en tiempo real cuando se supere un tamaño mediante la propia configuración de ***rsyslog.*** Todo ello mediante los ***output channels.***

Para entender cómo funcionaría se muestra un ejemplo para un fichero de ***log*** llamado ***/var/log/ prueba*** que nunca debería superar los ***12 MiB.*** Para ***rsyslog*** se creará un fichero de configuración / ***etc/rsyslog.d/prueba.conf*** con el siguiente contenido.

```
$outchannel        prueba,/var/log/prueba,12582912,/usr/sbin/logrotate        /
etc/logrotate.d/prueba
local5.info     $prueba
```

Para ***logrotate*** se creará el fichero de configuración ***/etc/logrotate.d/prueba.***

```
/var/log/prueba {
 weekly
 size 11M
 rotate 4
 compress
 delaycompress
 create 640 root adm
}
```

Es decir, cuando ***rsyslog*** escriba ***12 MiB*** en el fichero de log se lanzará ***logrotate*** utilizando la configuración ***/etc/logrotate.d/prueba.*** Con la configuración establecida el fichero de log será rotado si el tamaño alcanza más de ***11 MiB.*** El resultado obtenido será el siguiente.

```
# ls -alh /var/log/prueba*
-rw-r----- 1 root adm 4,9M mar 10 15:52 /var/log/prueba
-rw-r----- 1 root adm  12M mar 10 15:52 /var/log/prueba.1
-rw-r----- 1 root adm  41K mar 10 15:25 /var/log/prueba.2.gz
```

Por lo tanto ya se ha conseguido que los ***logs*** vayan rotando conforme a una serie de características, incluso pudiendo ser controlando el tamaño en tiempo real según esta última configuración. Pero, ¿es cómoda la revisión de ***logs*** por cada máquina? Incluso en entornos pequeños puede resultar pesado e inabordable, por lo que resulta necesario centralizar los ***logs.***

4. Logging remoto o centralizado

Una de las características esenciales en los sistemas de ***logging*** como ***rsyslog*** es sin duda la posibilidad de poder enviar los eventos hacia otra máquina de forma remota y en tiempo real. Esto permitirá crear un sistema centralizado de ***logs.***

Utilizar esta opción en ***rsyslog*** resulta muy sencillo, tan solo habilitando los módulos correspondientes para permitir conexiones remotas en la máquina que actúe como servidor de ***logging***, así como el envío de ***logs*** hacia la máquina remota desde el cliente.

Para demostrar el funcionamiento de ***logging*** remoto se dispone de un entorno constituido por dos máquinas.

- Máquina A. Actuará como servidor centralizado ***rsyslog.*** Se utilizará ***UDP*** para la recepción de mensajes utilizando el puerto estándar, ***514.*** La dirección IP de esta máquina es ***192.168.87.55.*** El ***hostname*** de esta máquina es ***test.***
- Máquina B. Será la máquina configurada como cliente remoto ***rsyslog.*** La dirección IP en este caso es ***192.168.87.53.*** El ***hostname*** para esta máquina es ***squeetest.***

Configuración de la máquina A

En el fichero de configuración por defecto ***/etc/rsyslog.conf*** aparecen comentadas las líneas que proporcionan la funcionalidad de recepción de ***logs.*** Una vez descomentadas quedarían del siguiente modo.

```
$ModLoad imudp
$UDPServerRun 514
```

Tras reiniciar el servicio ***rsyslog*** debería existir un socket de escucha para las conexiones ***rsyslog*** entrantes. Puede comprobarse mediante el comando ***netstat.***

```
# netstat -puna
...
udp    0      0 0.0.0.0:514         0.0.0.0:*               19339/rsyslogd
...
```

Ahora que la máquina A está lista para recibir ficheros de ***log*** llega el tumo de configurar la máquina B cliente.

Configuración de la máquina B

La configuración de la máquina cliente dependerá de los ***logs*** que deseen enviarse al sistema remoto, atendiendo a sus ***facilities*** por ejemplo. Un caso típico de configuración es el envío de todos

los mensajes a la máquina remota. Para ese caso debería agregarse la siguiente línea al fichero / ***etc/rsyslog.conf*** acuerdo al manual de ***rsyslog.***

```
*.*     @192.168.87.55
```

Es decir, se enviarán los eventos de todas las ***facilities*** a la máquina remota ***192.168.87.55.*** Por supuesto en la máquina local seguirán almacenándose los ficheros de acuerdo a la configuración establecida. Una vez reiniciado ***rsyslogd*** podría verificarse el funcionamiento utilizando el comando ***logger*** en la máquina B.

```
squeetest:~# logger -t Logger -p local1.info "Logging. Último capítulo."
squeetest:~# tail /var/log/syslog
...
Mar 10 18:10:45 squeetest Logger: Logging. Último capítulo.
```

En el caso de la máquina A, al comprobar el fichero donde se almacenen los eventos de la ***facility locall*** se obtendrá siguiente resultado. En este caso dicha ***facility*** se almacena en ***/var/log/syslog.***

```
root@test:~# tail -f /var/log/syslog
...
Mar 10 18:10:45 squeetest Logger: Logging. Último capítulo.
```

Sin embargo, uno de los problemas que presenta este método de almacenamiento remoto es la inseguridad proporcionada en la transmisión de los mensajes. Si por ejemplo un atacante consiguiera un entorno ***MiTM*** podría obtener información sensible de las máquinas. Una salida del comando ***tcpdump*** en una máquina atacante mostraría el siguiente mensaje.

```
192.168.87.53.60506 > 192.168.87.55.514: [udp sum ok] SYSLOG, length: 66
 Facility local1 (17), Severity info (6)
 Msg: Mar 10 18:10:15 squeetest Logger: Logging. \0xc3\0x9altimo cap\0xc3\0xadtulo.
  0x0000:  3c31 3432 3e4d 6172 2031 3020 3138 3a31
  0x0010:  353a 3039 2073 7175 6565 7465 7374 204c
  0x0020:  6f67 6765 723a 204c 6f67 6769 6e67 2e20
  0x0030:  c39a 6c74 696d 6f20 6361 70c3 ad74 756c
  0x0040:  6f2e
```

Como se puede observar toda la información viaja en claro. Para evitar este problema es posible configurar ***rsyslog*** para el envío de mensajes utilizando cifrado mediante ***TLS*** e incluso ***GSS-API*** para una infraestructura con ***Kerberos.*** Para la ayuda en la configuración de ***rsyslog*** utilizando alguno de estos modos es recomendable leer su documentación en línea.

Otras configuraciones interesantes

Además de la configuración remota ya mencionada, con ***rsyslog*** es posible utilizar otras características y funcionalidades como las siguientes.

- ***RELP.*** Es un protocolo de transmisión remota por el cual trata de asegurarse la entrega de los mensajes hacia el ***host*** de ***logging*** remoto. Incluso utilizando ***TCP*** para la transmisión de los eventos es posible que en ocasiones se pierda información. ***RELP*** soluciona este problema.
- Almacenamiento en bases de datos. Es posible configurar ***rsyslog*** de modo que almacene la información de los eventos en un sistema gestor de bases de datos. De esta forma se podrá ser más eficiente a la hora de realizar búsquedas en los registros. Es necesario tener en cuenta que en determinados entornos con demasiada carga de información, este método puede suponer un impacto en el rendimiento del sistema si la transferencia a las bases de datos es realizada en tiempo real.
- Análisis de registros mediante interfaces gráficas. Existen herramientas web como ***LogAnalizer*** que permitirán la utilización de infraestructuras de ***logging*** en bases de datos para un mejor análisis de los eventos, aplicación de filtros, etc.
- Alertas con ***logcheck.*** En muchas ocasiones la gran mayoría de los mensajes registrados en los ***logs*** son irrelevantes para los administradores de sistemas. Con la finalidad de que siempre obtengan información importante de funcionamiento existen aplicaciones como ***logcheck***, que podrán enviar sumarios de ciertos tipos de eventos previamente configurados.

Para más información sobre ***rsyslogd*** y sus posibilidades de configuración se recomienda la visita a su sitio oficial www.rsyslog.com**.**

Indice alfabético

Símbolos

0-day 153

A

ACCEPT 52, 55, 57, 58, 59, 60, 61, 62, 65, 66, 67, 68, 190, 193, 194
ACK 15, 19, 21, 33, 70, 71, 78, 79, 80, 96, 108, 109, 110, 111, 112, 113, 115, 116, 117, 146, 190, 199, 214, 218, 230, 233, 241, 251, 261, 268, 278
ACL 18, 157, 162, 163, 164, 165, 169, 170, 171, 224
agente 97, 201, 204, 205, 206, 207, 208
AH 121, 123, 124, 127, 129, 130, 131, 132, 134, 135, 139, 278
apache 21, 98, 100, 147, 149, 151, 157, 185, 186, 188, 203, 211, 212, 213, 214, 218, 220, 221, 222, 223, 225, 226, 227, 228, 229, 230, 232, 254, 266, 267
apt 77, 81, 98, 102, 103, 125, 133, 147, 148, 149, 166, 182, 185, 186, 194, 195, 196, 197, 198, 199, 201, 211, 226, 235, 251
ARP 73, 79, 108, 109, 111, 112, 230, 231, 278
arpspoof 108, 109, 111, 231
ataque 15, 19, 21, 33, 71, 78, 79, 80, 96, 108, 109, 110, 111, 112, 113, 115, 116, 117, 146, 190, 199, 214, 218, 230, 241, 251, 261, 268, 278

B

BASE 146, 148, 149, 151, 152, 278
BIOS 27, 28

C

captura 79, 130, 139, 144, 145, 152, 217, 230, 259
certificado 88, 90, 91, 92, 93, 94, 95, 96, 135, 229
cifrado 17, 19, 21, 25, 32, 34, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 76, 78, 80, 83, 87, 121, 124, 135, 190, 191, 219, 229, 233, 234, 237, 241, 242, 244, 246, 255, 270
clave pública 45, 46, 47, 48, 81, 135, 190, 196, 197, 198, 234, 236, 240, 241, 242, 243, 244, 245, 248, 249

D

Debian 11, 31, 32, 33, 35, 36, 37, 39, 58, 59, 72, 73, 74, 77, 81, 98, 99, 108, 119, 125, 126, 133, 147, 148, 149, 152, 159, 160, 166, 167, 168, 169, 173, 174, 176, 179, 180, 182, 185, 192, 194, 195, 196, 197, 199, 201, 211, 212, 213, 214, 216, 217, 219, 220, 226, 228, 235, 262, 263, 265
Diffie-Hellman 87, 88, 90, 94, 125, 134, 135, 233, 234, 236
dig 110
DNS 59, 60, 62, 64, 65, 66, 68, 77, 78, 79, 86, 108, 109, 110, 112, 113, 198, 258
dnsspoof 109, 110
DoS 96, 107, 109
dpkg 148, 149, 194, 203, 206
DROP 52, 55, 57, 59, 60, 62, 63, 64, 69, 70, 71, 74, 114, 116, 190

E

ESP 121, 123, 124, 125, 134, 135, 145

F

facility 103, 263, 264, 270
Fail2ban 250, 251, 252, 253
FILTER 72, 192
firewall 18, 19, 20, 23, 24, 30, 56, 58, 59, 62, 63, 64, 69, 71, 72, 73, 119, 146, 189, 190, 191, 251, 256
fortificación 11, 13, 14, 15, 21, 27, 211, 233, 241
FORWARD 52, 53, 55, 60, 61, 62, 64, 66, 67, 68, 69, 70, 71, 72, 74, 189, 190
fstab 40, 43, 162, 182, 249
fwknop 190, 191, 192, 193, 194

G

GNU/*Linux* 11, 15, 18, 22, 34, 35, 72, 76, 235, 251
GPG 44, 45, 46, 47, 195
GRUB 28, 29, 30, 31, 32, 33

H

hardening 199
HIDS 20, 146, 200, 201, 202, 204, 205, 206, 207, 208, 209, 209
host 20, 83, 98, 100, 101, 102, 103, 105, 112, 116, 117, 135, 146, 168, 169, 187, 192, 200, 201, 215, 216, 221, 234, 236, 239, 241, 242, 245, 263, 271
HTTP 55, 59, 60, 61, 62, 64, 65, 66, 68, 185, 190, 213, 217, 221, 222, 228, 229, 230, 231, 232, 253, 254, 255
HTTPS 59, 60, 61, 64, 65, 66, 68, 81, 213, 228, 229, 230, 231, 232

I

Icinga 97, 98, 99, 100, 102, 103, 104, 185
IDS 19, 20, 145, 146, 152, 201, 226, 244
IKE 121, 125, 129, 133, 134, 139, 141, 145
iptables 18, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 75, 83, 87, 96, 114, 115, 116, 120, 190, 191, 192, 193, 231, 251

L

LAMP 147, 148, 211, 217, 219
LFI 211
LOG 223, 225
logcheck 271
Login.defs 177
logrotate 265, 266, 267, 268
LUKS 34, 35, 37, 39, 40, 41, 42, 43, 44

M

MITM 78, 79, 107, 108, 109, 111
mod_security 226, 227, 228
Monit 185, 186, 188, 268
MSCHAPv2 76, 78, 79, 80
MySQL 147, 148, 149, 211, 212, 213, 214, 215, 216

N

Nagios 97, 98, 102
NAT 53, 62, 63, 68, 69, 73, 81, 117, 125, 145, 146
netfilter 51, 53, 69, 72
NIDS 20, 146, 147, 200

O

Open Source 97, 98
OpenVPN 76, 80, 81, 82, 83, 84, 86, 87, 88, 90, 91, 92, 94, 95, 96
OSSEC 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210

P

PAM 87, 91, 173, 174, 176, 177, 181, 203
passwd 29, 30, 87, 157, 158, 159, 160, 171, 174, 175, 178, 181, 215, 217, 218, 219, 208, 209

password 31, 32, 77, 78, 80, 89, 92, 93, 148, 159, 166, 167, 168, 169, 170, 171, 172, 173, 175, 177, 178, 179, 180, 181, 191, 193, 211, 216, 231, 245, 208
permiso 14, 22, 29, 50, 83, 126, 133, 153, 154, 157, 158, 159, 160, 163, 164, 165, 166, 167, 170, 172, 180, 200, 212, 216, 258, 263, 266
PHP 147, 211, 212, 213, 214, 216, 217, 218, 219, 220, 221
poisoning 108, 111
Port-Knocking 188, 189, 190, 191, 192
PPTP 76, 77, 78, 79, 80
protocolo 51, 55, 69, 74, 76, 79, 80, 87, 97, 116, 119, 120, 121, 122, 123, 124, 125, 127, 129, 131, 133, 139, 230, 233, 234, 235, 237, 238, 239, 241, 244, 246, 248, 250, 251, 253, 256, 257, 271
PSK 133, 134, 204, 205, 207

R

RFI 211, 218, 219
Rogue DHCP 113, 115, 116
RSA 48, 88, 89, 93, 197, 233, 234, 236, 239, 242, 243
rsyslog 99, 261, 262, 263, 264, 265, 266, 268, 269, 270, 271

S

SAD 121, 122, 125, 126, 127, 128, 129, 132, 136, 138, 139, 141, 144
SCP 28, 246, 247, 248
security 120, 121, 122, 124, 126, 127, 129, 131, 132, 136, 142, 143, 146, 174, 176, 178, 179, 194, 198, 203, 213, 218, 222, 226, 227, 228, 232
SFTP 248
shadow 29, 87, 159, 174, 176, 177, 180, 210
Snort 145, 146, 147, 148, 149, 150, 151, 152, 200, 226
SPA 189, 190, 191, 192, 193
SPD 122, 125, 126, 127, 128, 131, 132, 136, 138, 142
spoofing 79, 107, 112, 113, 114, 116, 198, 230, 231, 239
SQLi 211, 214, 215, 226, 228
SSH 19, 21, 55, 59, 61, 63, 64, 65, 66, 102, 186, 187, 188, 189, 190, 191, 192, 193, 194, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 244, 245, 246, 247, 248, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259
SSL 19, 228, 229, 230, 231
sslstrip 231
Sudo 199
Suhosin 219
SYN 70, 71, 189, 233

T

TCP 69, 70, 71, 76, 96, 124, 131, 185, 233, 244, 256, 271
TLS 76, 87, 92, 96, 228, 270
túnel 19, 76, 85, 86, 92, 95, 121, 122, 124, 127, 141, 142, 144, 145, 234, 246, 249, 253, 254, 255, 256, 258, 259

U

UDP 69, 96, 116, 124, 125, 131, 185, 257, 258, 269

V

VLAN 19, 116, 118, 119, 120
VPN 18, 19, 75, 76, 77, 78, 79, 80, 81, 82, 83, 85, 86, 91, 92, 95, 134, 145

W

Wireshark 128, 129, 130, 132, 139, 141, 144, 145, 217, 221, 230, 232, 259

X

XSS 211, 226