

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN
UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Proyecto Fin de Carrera

**Guía práctica para el manejo de herramientas de
seguridad para laboratorios docentes**



AUTOR: ANDRÉS FRANCISCO MUÑOZ GALLEGO

DIRECTOR: MARÍA DOLORES CANO BAÑOS

Octubre / 2012



Titulación	Ingeniero Técnico de Telecomunicaciones, esp. Telemática
Directora	María Dolores Cano Baños
Alumno	Andrés Francisco Muñoz Gallego
D.N.I.	23.030.234-N
Fecha de inicio	
Objetivos	<p>El planteamiento inicial surge por la necesidad de introducir algunos puntos de vista, que existen, a la hora de poder defender nuestro ordenador de Internet, debido a que la mayoría de las personas no tienen conocimiento de que Internet es un arma de doble filo. Entre los datos a proteger nos encontraremos con documentos, archivos, y datos personales o profesionales.</p> <p>Es esto lo que nos lleva a explicar de manera guiada algunos de los posibles métodos que pueden usarse para proteger o mejorar la seguridad en las redes de comunicación, incluyendo también explicaciones de cómo realizar ataques con el fin de demostrar lo sencillo que puede resultar romper la seguridad de nuestras contraseñas.</p>

Departamento	Tecnologías de la Información y las Comunicaciones
Curso académico	2012 / 2013
Título del Proyecto:	Guía práctica para el manejo de herramientas de seguridad para laboratorios docentes.

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi directora de proyecto María Dolores Cano Baños, que gracias a su dedicación y entusiasmo no hubiese sido posible realizar este trabajo.

A todos mis amigos que verán cumplido su deseo de invitarles por fin a una copa.

Y especialmente a toda mi familia (Padre, Madre, Sister, Abuela y Abuelo), que con su fuerza, motivación y alegría, he visto finalizada mi carrera, y en especial mención a mi abuelo, que donde quieras que estés, sé que has estado y estarás orgulloso de tu nieto.

A Silvia, la mujer que comparte su vida conmigo durante estos casi tres maravillosos años, y por su confianza, ya que sin ella esto no hubiese sido posible de conseguirlo.

A todos vosotros, muchas gracias.

INDICE

Capitulo 1. INTRODUCCIÓN.....	1
1.1 Objetivos.....	1
1.2 Estructura y descripción del proyecto.....	2
Capitulo 2. FORTALEZ DE LAS CONTRASEÑAS.....	3
2.1 Introducción.....	3
2.2 Pdftcrack.....	4
2.2.1 Pdftcrack para Windows.....	5
2.2.2 Pdftcrack para Linux.....	8
2.3 John the Ripper.....	9
2.3.1 John the Ripper para Windows.....	10
2.3.2 John the Ripper para Ubuntu.....	12
2.4 Ataque al WEP con Wifislax2.0.....	14
Capitulo 3. INGENIERÍA SOCIAL: PHISHING.....	18
3.1 ¿Qué es el Phishing?.....	18
3.2 Técnicas usadas en Phishing.....	18
3.3 Instalación del Servidor para el Phishing.....	19
3.4 Puesta en marcha del Phishing.....	20
Capitulo 4. ESCÁNERES DE REDES.....	25
4.1 Introducción.....	25
4.2 Nmap.....	27
4.2.1 Características Nmap.....	27
4.2.2 Funcionamiento Nmap.....	27
4.3 Nessus.....	30

Capítulo 5. AUTENTICACIÓN. RADIUS Y EAP.....	39
5.1 Introducción.....	39
5.1.1 Radius (Remote Authentication Dial – In Users Service).....	39
5.1.2 EAPOL (Extensible Authentication Protocol over LAN).....	40
5.1.2.1 Modo Operación.....	41
5.2 Desarrollo de las Instalaciones.....	42
5.2.1 Instalación del Servidor Radius.....	42
5.2.2 Configuración del Suplicante.....	44
5.2.3 Configuración del Autenticador.....	45
Capítulo 6. CERTIFICADOS DIGITALES PARA SERVIDORES WEB.....	51
6.1 Introducción.....	51
6.1.1 Certificados Digitales.....	51
6.1.2 Servidor Apache.....	53
6.2 Instalación Servidor Web Apache + Mod SSL.....	54
6.3 Instalación del Certificado Verisign.....	62
Capítulo 7. INSTALACIÓN Y MANEJO DE PGP.....	66
7.1 Introducción.....	66
7.2 Funciones y Servicios PGP.....	67
7.3 Descripción.....	68
7.3.1 Notación.....	68
7.3.2 Firma Digital o Autenticación.....	68
7.3.3 Confidencialidad.....	69
7.3.4 Autenticación y Confidencialidad.....	70
7.3.5 Compresión.....	71
7.3.6 Confidencialidad con correo electrónico.....	71
7.3.7 Segmentación.....	72
7.4 Gestión de claves.....	72

7.5	Instalación.....	74
7.5.1	Gestión clave Pública / Privada.....	76
7.6	Configuración.....	80
7.7	Funcionamiento.....	82
7.7.1	Funcionamiento para cifrar ficheros.....	82
7.7.2	Funcionamiento con el programa de correo Outlook Express.....	83
Capítulo 8. CONCLUSIONES.....		85
BIBLIOGRAFÍA.....		86



CAPITULO 1. Introducción

1.1. Objetivos

El planteamiento inicial surge por la necesidad de introducir algunos puntos de vista, que existen, a la hora de poder defender nuestro ordenador de Internet, debido a que la mayoría de las personas no tienen conocimiento de que Internet es un arma de doble filo. Entre los datos a proteger nos encontraremos con documentos, archivos, datos personales e incluso profesionales.

Es esto lo que nos lleva a explicar de manera guiada algunos de los posibles métodos que pueden usarse para proteger o mejorar la seguridad en las redes de comunicación, incluyendo también explicaciones de cómo realizar ataques con el fin de demostrar lo sencillo que puede resultar romper la seguridad de nuestras contraseñas.

Para la consecución de este objetivo, el proyecto consta de las siguientes fases (preparación, instalación y guía práctica de):

1. Fortaleza de las contraseñas. Ataque por diccionario y ataque a WEP.
2. Ataque tipo *phishing*.
3. Escáneres de redes.
4. Autenticación. RADIUS y EAP.
5. Certificados digitales para servidores WEB.
6. Instalación y manejo de PGP.

1.2. Estructura y descripción del proyecto

La estructura de este proyecto está formada por los siguientes capítulos:

- ❖ **Capítulo 2.** Se explica la fortaleza de las contraseñas. Aquí usamos distintas herramientas para comprobar el nivel de seguridad de nuestras contraseñas usando distintos tipos de ataques: Ataque por fuerza bruta, por diccionario y ataque al WEP.
- ❖ **Capítulo 3.** Se desarrolla la Ingeniería Social mediante un ataque tipo *phishing*, en la que mostramos un ejemplo práctico, ya que este es uno de los ataques más comunes que podemos sufrir y las consecuencias pueden llegar a ser importantes sino somos conscientes a las paginas *webs* que accedemos y donde introducimos nuestros datos personales, para ello únicamente hemos descargado a nuestro ordenador un servidor Apache, configurándolo adecuadamente y realizando el ataque.
- ❖ **Capítulo 4.** Realizaremos un escáneres de redes, el objetivo principal es conocer el uso de los escáneres de redes, en particular, las aplicaciones *Nmap* y *Nessus* tanto



configuración, ejecución e instalación. Aquí realizamos una aplicación que permite realizar una verificación de seguridad en una red de forma automática mediante el análisis de los puertos abiertos en uno de los equipos o en toda la red. El proceso de análisis utiliza sondas (solicitudes) que permiten determinar los servicios que se están ejecutando en un *host* remoto así como los niveles de riesgo en la que se encuentran cada uno de los *hosts*.

- ❖ **Capítulo 5.** Realizaremos una autenticación mediante *RADIUS* y *EAP*. El objetivo principal es conocer de forma sencilla el funcionamiento del sistema de autenticación *EAP* con *RADIUS* (mensajes que se intercambian, etc...), mediante la configuración de un servidor *RADIUS*, un cliente *EAP* y un *switch* para ser utilizado como autenticador en un sistema de autenticación.
- ❖ **Capítulo 6.** Uso de certificados digitales para servidores *WEB*. En esta sección aprenderemos a instalar un servidor *WEB* con facilidades criptográficas, solicitar e instalar un certificado X.509 firmado por una Autoridad Certificadora.
- ❖ **Capítulo 7.** Instalación y manejo de *PGP* consiste en familiarizarse con la aplicación software de libre distribución *PGPTools* con la que se realizarán distintas actividades destinadas a la creación de anillos de confianza, el cifrado de archivos y la firma de mensajes.



CAPITULO 2. Fortaleza de las Contraseñas

2.1. Introducción

En este capítulo, mostraremos algunas de las vulnerabilidades que puede presentar un equipo (PC, móvil, etc.), en cuestión de contraseñas, ya sea bien por la dejadez o por el desconocimiento, lo que a veces nos hace usar contraseñas tan débiles como pueden ser nuestra fecha de nacimiento, número de D.N.I., o incluso nuestro propio nombre.

Para ello usamos distintas herramientas para comprobar el nivel de seguridad de nuestras contraseñas mediante distintos tipos de ataques:

- ❖ Ataque por diccionario, en este tipo de ataque utilizamos distintas herramientas de libre distribución para diferentes plataformas (*Unix, Win's...*), con el objetivo principal de realizar de manera guiada ejemplos prácticos de cómo se realizan ataques a nuestros archivos de contraseñas, tanto a nivel de documentos como de sistema.
- ❖ Ataque por fuerza bruta, es la forma de recuperar una clave probando todas las combinaciones posibles hasta encontrar aquella que permite el acceso.
- ❖ Ataque al *WEP*, aquí mostramos de manera explícita lo vulnerable que resulta romper una contraseña, independientemente del tamaño de la clave, ya que, a mayor tamaño el tiempo de ruptura se vendría aumentado pero con el mismo éxito que nos propusimos, únicamente con el uso de un programa

Las herramientas que hemos utilizado para realizar los diferentes ataques, tanto para sistemas operativos Unix como para Windows, son lo que aparece en la *Tabla 1*.

Herramientas	Tipos de ataque
PdfCrack	- Ataque por Diccionario - Ataque por Fuerza Bruta
John The Ripper	- Ataque por Diccionario - Ataque por Fuerza Bruta
Wifislax2.0	- Ataque al WEP

Tabla 1. Herramientas para ataques a contraseñas.



Las descripciones de las distintas herramientas son:

- ❖ **PDFcrack:** herramienta *GNU/Linux* diseñada para la recuperación de contraseñas y contenido de ficheros de extensión *pdf*. Dicha herramienta es pequeña y se usa en línea de comandos (o terminal) de manera muy sencilla. La aplicación es *OpenSource (GPL)*.
- ❖ **John the Ripper:** es una herramienta para descifrar contraseñas tanto en *Windows* y *Unix*. Utiliza algoritmos propios. Admite muchas reglas y opciones y está bien documentada.
- ❖ **Wifislax:** es un programa de distribución en *Linux* creada desde España y enfocada en la seguridad *WiFi*.

2.2. Pdfcrack

Pdfcrack es una de las herramientas más potentes que tenemos para la recuperación de documentos *PDF*'s protegidos con contraseñas. Funciona sobre línea de comandos de sistemas *Linux* y *Windows*, basado en el ataque por fuerza bruta y ataque por diccionario.

Donde las opciones de la pantalla principal para los dos sistemas operativos son:

- ❖ - B: Realizar evaluación comparativa y salir.
- ❖ - C [*Cadena*]: Utiliza los caracteres indicados para la búsqueda de la contraseña.
- ❖ - m [*Integer*]: La búsqueda para llegar a la longitud de *INTEGER*.
- ❖ - n [*Integer*]: No se probarán las contraseñas menores en caracteres que las indicadas en *INTEGER*.
- ❖ - l [*File*]: Continuar desde el punto guardado en el fichero *FILE*.
- ❖ -q: Ejecutar silenciosamente.
- ❖ -s: Realizar la búsqueda de la contraseña permutando las mismas.
- ❖ - u: Trabajar con el "userpassword".
- ❖ - v: Imprime la versión.
- ❖ - w [*File*]: Usa la lista dentro de *FILE* como fuente de contraseñas para la búsqueda.

El uso del programa *Pdfcrack* es sencillo. Basta con poner la opción `-f` seguido del nombre del fichero *pdf* y cuya clave hemos olvidado, realizando un ataque por fuerza bruta, mediante el comando:

```
pdfcrack -f archivo_con_clave.pdf
```

Con el objetivo de reducir el tiempo de ejecución, se podría usar las siguientes opciones (algunas de ellas mencionadas anteriormente) como muestra la *Tabla 2*.



Charset = CHARSET	Prueba todas las combinaciones de caracteres indicados en CHARSET, existiendo diferencias ya que contraseñas escritas en minúsculas es totalmente diferente a contraseñas escritas en mayúsculas.
Maxpw = INTEGER	La longitud máxima para las claves es INTEGER.
Minpw = INTEGER	La longitud mínima para las claves es INTEGER.
Wordlist = FILE	Usa el fichero FILE como un diccionario de palabras a probar.

Tabla 2. Opciones Pdfrack.

2.2.1. Pdfrack para Windows

En la *Figura 1*, se puede observar las opciones del programa en el sistema operativo *Windows*.

```
c:\ Símbolo del sistema
D:\Archivos de programa>cd Pdfrack
D:\Archivos de programa\Pdfrack>pdfrack.exe
Usage: pdfrack -f filename [OPTIONS]
OPTIONS:
-b, --bench          perform benchmark and exit
-c, --charset=STRING Use the characters in STRING as charset
-w, --wordlist=FILE  Use FILE as source of passwords to try
-n, --minpw=INTEGER  Skip trying passwords shorter than this
-m, --maxpw=INTEGER  Stop when reaching this passwordlength
-l, --loadState=FILE Continue from the state saved in FILENAME
-o, --owner          Work with the ownerpassword
-u, --user          Work with the userpassword (default)
-p, --password=STRING Give userpassword to speed up breaking
                    ownerpassword (implies -o)
-q, --quiet         Run quietly
-s, --permutate     Try permutating the passwords (currently only
                    supports switching first character to uppercase)
-v, --version       Print version and exit
D:\Archivos de programa\Pdfrack>
```

Figura 1. Menú principal Pdfrack en Windows.

Ahora mostraremos una comparación de los tiempos que hemos tomado a la hora de utilizar distintas opciones de *Pdfrack* como se muestra en la *Tabla 3* y *Tabla 4*.



CONTRASEÑA	OPCIÓN	TIEMPO
Sito	<i>Pdftcrack -f</i>	10 segundos
	<i>Pdftcrack --minpw=3</i>	7 segundos
	<i>Pdftcrack -s</i>	9 segundos

Tabla 3. Tiempo y opciones ataque Pdftcrack.

```
c:\> Símbolo del sistema - pdftcrack -f "Ejemplo Pdftcrack1.protected.pdf"
D:\Archivos de programa\Pdftcrack>pdftcrack -f "Ejemplo Pdftcrack1.protected.pdf"
PDF version 1.5
Security Handler: Standard
U: 1
R: 2
P: -4
Length: 40
Encrypted Metadata: True
FileID: 19fb19e971435460a733d3b6931482cc
U: 88414b915de459d4790c2dc397bf54068d89081b47bf4582ac03f9a36433dfe0
O: 041542040b225c18ade70150370fba3c28abbf83cde279283bc42587c7460fab
found user-password: 'sito'
```

Figura 2. Opción pdftcrack -f.

```
c:\> Símbolo del sistema - pdftcrack --minpw=3 "Ejemplo Pdftcrack1.protected.pdf"
D:\Archivos de programa\Pdftcrack>pdftcrack --minpw=3 "Ejemplo Pdftcrack1.protected
.pdf"
PDF version 1.5
Security Handler: Standard
U: 1
R: 2
P: -4
Length: 40
Encrypted Metadata: True
FileID: 19fb19e971435460a733d3b6931482cc
U: 88414b915de459d4790c2dc397bf54068d89081b47bf4582ac03f9a36433dfe0
O: 041542040b225c18ade70150370fba3c28abbf83cde279283bc42587c7460fab
found user-password: 'sito'
```

Figura 3. Opción pdftcrack -- minpw.

```
c:\> Símbolo del sistema - pdftcrack -s "Ejemplo Pdftcrack1.protected.pdf"
D:\Archivos de programa\Pdftcrack>pdftcrack -s "Ejemplo Pdftcrack1.protected.pdf"
PDF version 1.5
Security Handler: Standard
U: 1
R: 2
P: -4
Length: 40
Encrypted Metadata: True
FileID: 19fb19e971435460a733d3b6931482cc
U: 88414b915de459d4790c2dc397bf54068d89081b47bf4582ac03f9a36433dfe0
O: 041542040b225c18ade70150370fba3c28abbf83cde279283bc42587c7460fab
found user-password: 'sito'
```

Figura 4. Opción pdftcrack -s.



CONTRASEÑA	OPCIÓN	TIEMPO
Sito123	<i>Pdfcrack -f</i>	2 días
	<i>Pdfcrack --minpw=4</i>	Aprox. 32 horas
	<i>Pdfcrack --s</i>	Aprox. 26 horas

Tabla 4. Tiempo y opciones ataque Pdfcrack.

```

C:\> Símbolo del sistema - pdfcrack -f "Ejemplo Pdfcrack2.protected.pdf"

D:\Archivos de programa\pdfcrack>pdfcrack -f "Ejemplo Pdfcrack2.protected.pdf"
PDF version 1.5
Security Handler: Standard
U: 2
R: 3
P: -4
Length: 128
Encrypted Metadata: True
FileID: ddc3e99dc4a12372409938ecdb9562d1
U: 7a77aa7e16316c547b646319e8f72502000000000000000000000000000000000000
O: 298136a81d5264d309a9c5abf7023dbeb5b3c02cebea4e13b51d9a6d9678e30d
Average Speed: 10246.8 w/s. Current Word: 'As0'
Average Speed: 10240.0 w/s. Current Word: 'NJRa'
Average Speed: 9602.5 w/s. Current Word: 'oHFb'
Average Speed: 7439.6 w/s. Current Word: 'gpic'

.....

Average Speed: 6921.0 w/s. Current Word: 'tio3122'
Average Speed: 7017.7 w/s. Current Word: 'ii34io3'
Average Speed: 6906.6 w/s. Current Word: 'It2ot13'
found user-password: 'sito123'
  
```

Figura 5. Opción pdfcrack -f.

```

C:\> Símbolo del sistema - pdfcrack --minpw=4 "Ejemplo Pdfcrack2.protected.pdf"

D:\Archivos de programa\pdfcrack>pdfcrack --minpw=4 "Ejemplo Pdfcrack2.protected
.pdf"
PDF version 1.5
Security Handler: Standard
U: 2
R: 3
P: -4
Length: 128
Encrypted Metadata: True
FileID: ddc3e99dc4a12372409938ecdb9562d1
U: 7a77aa7e16316c547b646319e8f72502000000000000000000000000000000000000
O: 298136a81d5264d309a9c5abf7023dbeb5b3c02cebea4e13b51d9a6d9678e30d
Average Speed: 11416.8 w/s. Current Word: '0y7a'
Average Speed: 8499.8 w/s. Current Word: 'SMPb'
Average Speed: 9229.2 w/s. Current Word: '2NBc'
Average Speed: 8100.6 w/s. Current Word: '7Whd'

.....

Average Speed: 5303.9 w/s. Current Word: 'N5lbxa'
Average Speed: 8125.5 w/s. Current Word: 'K9rdxa'
Average Speed: 5816.4 w/s. Current Word: 'rkcfxa'
Average Speed: 5405.1 w/s. Current Word: 'Kwugxa'
Average Speed: 6812.8 w/s. Current Word: 'bkoixa'
Average Speed: 6927.1 w/s. Current Word: 'Gmjksa'
Average Speed: 6564.1 w/s. Current Word: 'm4amxa'
Average Speed: 6961.0 w/s. Current Word: 'vb6nxa'

.....

found user-password: 'sito123'
  
```

Figura 6. Opción pdfcrack --minpw.



Como se puede observar, mediante las opciones que te proporciona la herramienta *Pdfcrack* se puede disminuir considerablemente los tiempo de descifrado de la contraseña a través del ataque por fuerza bruta.

2.2.2. Pdfcrack para Linux

En la *Figura 7* se puede observar las opciones del programa Pdfcrack en el sistema operativo Linux.

```
sito@sito-laptop:~$ pdfcrack
Usage: pdfcrack -f filename [OPTIONS]
OPTIONS:
-b, --bench           perform benchmark and exit
-c, --charset=STRING Use the characters in STRING as charset
-w, --wordlist=FILE   Use FILE as source of passwords to try
-n, --minpw=INTEGER  Skip trying passwords shorter than this
-m, --maxpw=INTEGER  Stop when reaching this passwordlength
-l, --loadState=FILE Continue from the state saved in FILENAME
-o, --owner           Work with the ownerpassword
-u, --user            Work with the userpassword (default)
-p, --password=STRING Give userpassword to speed up breaking
                    ownerpassword (implies -o)
-q, --quiet          Run quietly
-s, --permutate      Try permutating the passwords (currently only
                    supports switching first character to uppercase)
-v, --version        Print version and exit
sito@sito-laptop:~$
```

Figura 7. Menú principal Pdfcrack en Linux.

En el ejemplo que hemos realizado, consta básicamente de a partir de una relación de posibles palabras que hemos diseñado mediante la **opción** *wordlist*, el programa realizara las comprobaciones en base a estas posibles contraseñas. Para ello ejecutamos el comando:

```
pdfcrack--wordlist=@_del_diccionario @_de_archivo_pdf_con_contraseña
```

Tras ejecutar el programa, vemos que nuestro documento pdf estaba cifrado con la contraseña **“andito”** mostrada en la *Figura 8*.



```

sito@sito-laptop:~$ pdfcrack
Usage: pdfcrack -f filename [OPTIONS]
OPTIONS:
-b, --bench           perform benchmark and exit
-c, --charset=STRING  Use the characters in STRING as charset
-w, --wordlist=FILE   Use FILE as source of passwords to try
-n, --minpw=INTEGER  Skip trying passwords shorter than this
-m, --maxpw=INTEGER  Stop when reaching this passwordlength
-l, --loadstate=FILE  Continue from the state saved in FILENAME
-o, --owner           Work with the ownerpassword
-u, --user            Work with the userpassword (default)
-p, --password=STRING Give userpassword to speed up breaking
                    ownerpassword (implies -o)
-q, --quiet          Run quietly
-s, --permutate      Try permutating the passwords (currently only
                    supports switching first character to uppercase)
-v, --version        Print version and exit
sito@sito-laptop:~$ pdfcrack --wordlist=/home/sito/Escritorio/spanish /home/sito/Escritorio/prueba1.pdf
PDF version 1.4
Security Handler: Standard
V: 2
R: 3
P: -1028
Length: 128
Encrypted Metadata: True
FileID: 7d5a52eba4836b08c217e454bd8989e2
U: 3123b7774784c46193755a6a50069b4c000000000000000000000000000000
O: 199fc09b65b1550941a7eead70c57d7cb2a2457c1bcf4a8dd0f4478391caa733
found user-password: 'andito'
sito@sito-laptop:~$

```

Figura 8. Contraseña recuperada con Pdftcrack.

2.3. John the Ripper

John the Ripper es un programa de criptografía que aplica ataques de diccionario y por fuerza bruta para descifrar contraseñas. Es capaz de romper varios algoritmos de cifrado o *hash*. Es una herramienta de seguridad muy popular, ya que permite a los administradores de sistemas comprobar que las contraseñas de los usuarios son lo suficientemente buenas. *John the Ripper* es capaz de detectar el tipo de cifrado de entre muchos disponibles, y se puede personalizar su algoritmo de prueba de contraseñas. Esto ha hecho que sea uno de los más usados en este campo.

Entre los **algoritmos** más destacados que entiende, se encuentran:

- ❖ DES, MD5
- ❖ Kerberos AFS
- ❖ Hash LM
- ❖ MD4
- ❖ LDAP
- ❖ MySQL

Las principales **características** que presenta son:

- ❖ Optimizado para muchos modelos de procesadores.
- ❖ Funciona en muchas arquitecturas y sistemas operativos.
- ❖ Ataques de diccionario y por fuerza bruta.
- ❖ Muy personalizable (es software libre).



- ❖ Permite definir el rango de letras que se usará para construir las palabras y las longitudes.
- ❖ Permite para el proceso y continuarlo más adelante.
- ❖ Permite incluir reglas en el diccionario para decir cómo han de hacerse las variaciones a la hora de la búsqueda de la contraseña.
- ❖ Se puede automatizar.

John the Ripper usa un ataque por diccionario: tiene un diccionario de palabras, que pueden ser contraseñas típicas, y las va probando todas. Para cada palabra, la cifra y la comprara con el *hash* a descifrar, esto funciona bien porque la mayor parte de las contraseñas que usan las personas son palabras de diccionario. Pero *John the Ripper* también prueba las variaciones de estas palabras, añadiendo números, signos mayúsculas y minúsculas, cambia letras, combina palabras, etc. Además ofrece un ataque de fuerza bruta en el que se prueban todas las combinaciones posibles, sean palabras o no. Éste es el sistema más lento, y usado sólo en casos concretos, ya que el ataque por diccionario permite descubrir muy rápidamente las contraseñas débiles.

En este apartado vamos a describir el funcionamiento de la herramienta *John the Ripper* para recuperación de contraseña (o ataque por contraseña).

En primer lugar veremos su funcionamiento con el sistema operativo *Windows* y posteriormente con sistema *Ubuntu*.

2.3.1. John The Ripper para Windows

Las contraseñas de los equipos en *Windows* no se guardan directamente en un archivo, por motivos de seguridad, ya que si fuera así, el acceso a ellas sería tan simple como conocer la dirección del archivo en el disco duro, algo bastante fácil para el que quisiera saberlas. Es por este motivo que estas se almacenan de manera cifrada. En *Windows* el sistema las cifra en un archivo *SAM*, que se encuentra en un lugar de la carpeta *SYSTEM*, y cuya ruta se encuentra en:

```
c:\WINDOWS\system32\config\SAM
```

Este archivo contiene los llamados *LM hashes* que están cifrados.

Para poder usar esta aplicación se requiere de otra herramienta de ayuda para extraer el *SAM* de *Windows*, donde se almacena las claves, esta herramienta es *Pwdump2*, su implementación es muy sencilla y similar al funcionamiento de *John The Ripper* ya que tenemos que usar la consola y ejecutarlo en el directorio donde está el programa (sólo funciona en línea de comandos), entonces poniendo este comando:

```
C:\Documnts and Settings\sito\rwdump2\rwdump2.exe>passwd.txt
```



El archivo “*passwd.txt*” no es más que el archivo donde vamos a copiar el SAM, con la siguiente información:

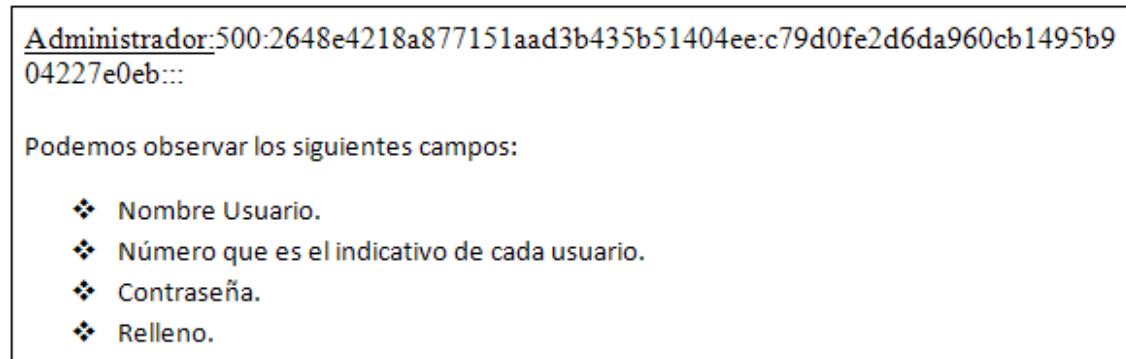


Figura 9. Archivo *passwd.txt*.

Ahora solo tenemos que copiar el archivo con extensión “*.txt*” que nos ha devuelto *pwdump2* y copiarlo en la carpeta *run* del programa *John The Ripper*:

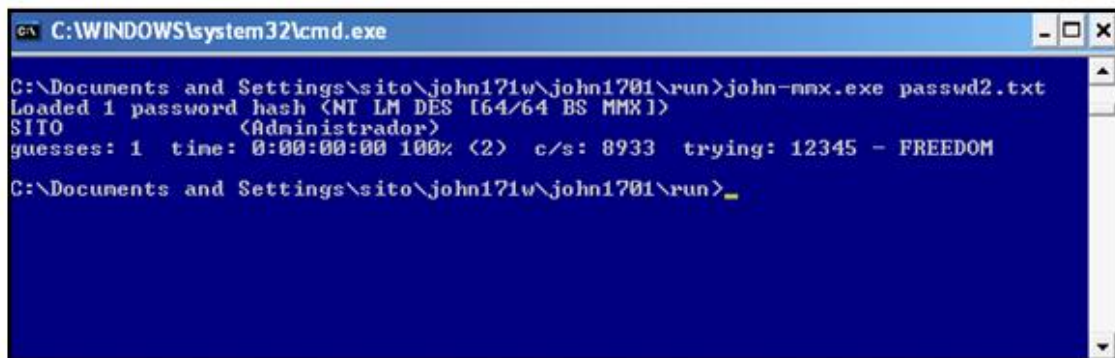
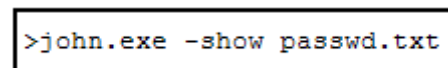


Figura 10. Obtención del password.

En la *Figura 10* observamos que el programa nos muestra por pantalla si ha conseguido la contraseña y vemos que en nuestro caso es “SITO”, el programa las almacenará en un archivo llamado “*john.pot*” y también nos la muestra con el comando:



Si no conseguimos obtenerla, podemos probar a configurar otros parámetros más avanzados con combinaciones, permutaciones o diccionarios gigantes, debido a que el programa *John The Ripper* en cuanto a comandos son los mismos tanto para los Sistemas Operativos *Unix* y *Windows*.



2.3.2. John The Ripper para Ubuntu

Ahora vamos a comentar las distintas posibilidades que nos ofrece este programa. La *Figura 11* muestra el aspecto general que presenta.

```
Archivo Editar Ver Terminal Solapas Ayuda
sito@sito-laptop:~$ john

John the Ripper Version 1.6 Copyright (c) 1996-98 by Solar Designer

Usage: john [OPTIONS] [PASSWORD-FILES]
-single                "single crack" mode
-wordfile:FILE -stdin  wordlist mode, read words from FILE or stdin
-rules                 enable rules for wordlist mode
-incremental[:MODE]   incremental mode [using section MODE]
-external:MODE         external mode or word filter
-stdout[:LENGTH]      no cracking, just write words to stdout
-restore[:FILE]        restore an interrupted session [from FILE]
-session:FILE          set session file name to FILE
-status[:FILE]         print status of a session [from FILE]
-makechars:FILE        make a charset, FILE will be overwritten
-show                  show cracked passwords
-test                  perform a benchmark
-users:[-]LOGIN|UID[,...] load this (these) user(s) only
-groups:[-]GID[,...]   load users of this (these) group(s) only
-shells:[-]SHELL[,...] load users with this (these) shell(s) only
-salts:[-]COUNT      load salts with at least COUNT passwords only
-format:NAME            force ciphertext format NAME (DES/BSDI/MD5/BF/AFS/LM)
-savemem:LEVEL         enable memory saving, at LEVEL 1..3

sito@sito-laptop:~$
```

Figura 11. Visión del menú principal John The Ripper.

Lo primero que tenemos que hacer para utilizar esta herramienta, es comprobar si las contraseñas que se encuentran en el archivo, están ocultas. El archivo es:

```
/etc/passwd
```

Para ello ejecutamos el comando:

```
more /etc/passwd
```

Comprobamos que no lo están, ya que nos sale la siguiente línea de comando:

```
root:BsDvGHY.FD:0:0:root:/root:/bin/bash
```

Vemos que si en lugar de "BsDvGHY.FD" nos encontráramos una "x" o "**", la contraseña se encuentra oculta, y más adelante se explicará qué debemos hacer cuando esto ocurra.

La **primera opción** que podemos usar con el programa *John the Ripper* es "single". La línea de comando sería:

```
>john -single passwd
```



Donde *passwd* es el nombre del archivo donde se encuentran las contraseñas (podría tener otro nombre).

Como **segunda opción** es mediante la línea de comando:

```
>john -wordlist:passwords.lst passwd
```

El comando *wordlist*, nos permite usar un listado de palabras que se ha obtenido de dos maneras posibles:

- ❖ Manualmente.
- ❖ Lista prediseñada.

La **tercera opción**, es el comando *rules* que nos permite poner a cada palabra variaciones del tipo sufijo o prefijo, lo que en algunos casos puede ahorrar tiempo. La línea de comando es:

```
>john -wordlist:passwords.lst passwd -rules
```

Como **cuarta opción** es el modo incremental, es más potente que el resto de opciones pero tiene el inconveniente del tiempo de espera, las opciones que nos encontramos se muestran en la *Tabla 5*:

alpha	Genera palabras con letras solamente (máximo 27 letras). <pre>>john -i:alpha passwd</pre>
digits	Genera palabras con números solamente (desde 0 hasta 9). <pre>>john -i:digits passwd</pre>
all	Genera palabras con letras, números y caracteres especiales (máximo 90 caracteres). <pre>>john -i:all passwd</pre>
lanman	Genera palabras con letras, números y algunos caracteres especiales. <pre>>john -i:lanman passwd</pre>

Tabla 5. Opciones modo Incremental.

La **quinta opción** es el comando *shadow*, aquí retomamos el tema del principio en relación a las contraseñas que se encuentran ocultas, en este caso la forma de actuar es la de encontrar la contraseña y el */etc/shadow* para posteriormente unirlos y poder trabajar con la



herramienta *John the Ripper*. Para ello lo primero es unir tanto la contraseña como el *shadow* y meterlo en un archivo, para posteriormente usar el archivo como hemos visto en las otras opciones, para ello tenemos dos opciones de unión:

1. La primera de ellas es uniéndolo e intentar descifrar la contraseña en un sólo comando como:

```
>john /etc/passwd /etc/shadow > nuevo-password
```

Donde:

```
/etc/passwd → es password-FILE  
/etc/shadow → es shadow-FILE  
nuevo-password → es el archivo unido con todas las contraseñas
```

2. La segunda sería uniéndola con el comando:

```
>unshadow /etc/passwd /etc/shadow>nuevo-password
```

y después intentamos encontrar la contraseña.

Cuando consigamos alguna contraseña, ésta es enviada a un archivo que se encuentra en el JTR llamado *john.pot*

2.4. Ataque al WEP con Wifislax 2.0

La herramienta **Wifislax** es sin lugar a dudas una de las mejores y más potente para la auditoria *Wireless*, cual nos ayudará aprobar la seguridad de nuestra red *Wifi*, y así evitar ser víctimas como está mucho de moda a su gran facilidad.

Para ello hemos utilizado una herramienta que no es más que un CD de arranque que contiene el sistema operativo *Linux*, aunque existe la posibilidad de poder instalarlo directamente a nuestro disco duro.

En la *Tabla 6* se muestra de que herramientas esta formado El *Wifislax2.0*:

Airdump	Nos muestra una lista de los puntos de acceso detectados y también una lista de los clientes conectados (<i>station</i>).
Aircrack	Puede recuperar la clave WEP una vez que se han capturado suficientes paquetes cifrados con <i>airdump</i> .



Aireplay	Genera tráfico para usarlo más tarde con aircrack y poder descifrar las claves WEP.
-----------------	---

Tabla 6. Herramientas Wifislax2.0.

Ahora vamos a realizar los pasos para realizar un ataque:

1. Comprobar que los **DRIVERS** se encuentran en el *shell*, para ello debemos escribir el siguiente comando:

```
dmesg | grep ipw
```

Para poder ver en que *ethx* aparece nuestra ipw2200 y si está activo rtap0 por defecto para poder capturar los paquetes, y para ello necesitamos introducir el siguiente comando:

```
iwconfig
```

2. Tenemos en la *Figura 11* el **ESCANEO DE REDES**, aquí verificamos todas las redes que están en nuestro alcance, para ello escribimos en el *shell*:

```
airodum-ng rtap0
```

```
CH 0 ][ Elapsed: 4 s ][ 2008-05-07 12:44
BSSID      PWR Beacons  #Data, #/s  CH  MB  ENC  CIPHER AUTH ESSID
00:14:7C:43:B7:DC  -1    58      203  48  11  54.  WEP  WEP   lama
BSSID      STATION    PWR  Lost  Packets  Probes
00:14:7C:43:B7:DC  00:14:A5:45:F5:4A  -1    2      119
00:14:7C:43:B7:DC  00:1B:77:10:05:38  -1    0       84
wifislax - #
wifislax - #
```

Figura 11. Escaneo de Redes.

Una vez que hemos visto esto, pausamos con “CONTROL + C”.

3. En este paso haremos la **CAPTURA**, iniciamos el *airodum-ng* de nuevo para capturar paquetes de una red específica, es decir, nuestro objetivo e ir guardándolos en un archivo. Utilizaremos *rtap0* para capturar ya que si utilizamos *ethx* ahora, luego no podremos inyectar con *ethx*. Para ello escribimos en un nuevo *shell*:

```
airodump-ng -c canal -b bssid -w nombre_de_la_captura rtap0
```



4. **ATAQUE**, es necesario que nuestra interfaz *ethx* esté en modo *managed*, ya que por defecto lo está, debido a que *Wifislax* trabaja en este modo y no en modo monitor. Mediante el comando:

```
iwconfig ethx
```

, y si está en modo monitor habría que poner el comando:

```
iwconfig ethx mode managed
```

5. **DESCIFRAR CLAVE**, si todo ha ido bien deberíamos de tener muchos paquetes en *#DATA*. Lo ideal es 250.000 paquetes capturados para una clave de 128bits, y el doble para unas más grande. En la *Figura 12* se muestra la clave descifrada.

Tenemos 3 opciones dependiendo de la versión que tenemos:

```
aircrack-ptw nombre_de_la_captura → Este es el más rápido
```

```
aircrack-ng nombre_de_la_captura → Menos rápido que el primero
```

```
aircrack nombre_de_la_captura → El más lento
```

```
aircrack 2.41
www.guerrillawireless.net

[00:00:05] Probadas 1 claves (hay 1251815 IVs)

KB profundidad byte(vote)
0 0/ 1 F1( 240) 7D( 15) C4( 0) 71( -7) 12( -14) 92( -17)
1 0/ 1 9D( 262) CF( 28) 06( 15) 7C( 1) 88( 0) AE( -15)
2 0/ 1 0C( 294) 64( 16) 57( 5) 2D( 3) 63( 3) 7B( 0)
3 0/ 1 04( 304) 52( 20) 1B( 13) 44( 12) E5( 6) 50( 3)

A clave encontrada! [ F1:9D:0C:D4:ED ]

wifislax ~ #
```

Figura 12. Clave descifrada.

Las capturas son guardadas y si no sabemos donde se guardan podemos hacer en el *shell* y nos lo muestra ver *Figura 13*.



```
wifislax - # ls
Desktop/   replay_arp-0506-133929.cap  sito-01.cap  sotano-01.cap  swireless/
bluetooth/ replay_arp-0507-124616.cap  sito-01.txt  sotano-01.txt
wifislax - #
```

Figura 13. Ubicación de las capturas obtenidas.



CAPITULO 3. Ingeniería social: Phishing

3.1. ¿Qué es el Phishing?

Actualmente es más habitual recibir mensajes de correo electrónico que se hacen pasar por comunicados de bancos, tiendas de internet o páginas de pago, con el fin de reclamar la atención de los destinatarios para actualizar sus claves de acceso o confirmar su número de tarjeta de crédito a través de un enlace que les conduce a páginas web falsas para suplantar nuestra identidad y poder realizar todo tipo de operaciones en Internet, este fraude se denomina *Phishing* y es posiblemente la modalidad de fraude más extendida en Internet, especialmente a través del correo electrónico.

3.2. Técnicas usadas en el phishing

El principal objetivo del phishing es el de conseguir todo tipo de cuentas de usuarios, que posteriormente son usadas de forma fraudulenta para el envío de *Spam* desde cuentas comprometidas, robo de dinero mediante acceso a servicios de banca electrónica o uso de tarjetas de crédito de las víctimas, etc.

El procedimiento usado básicamente es el siguiente:

- ❖ El usuario recibe un e-mail de un banco, entidad financiera, tienda de Internet, Universidad, entre otros, en el que se le explica que por motivos de seguridad, mantenimiento, mejora del servicio, confirmación de identidad o cualquier otro motivo, debe actualizar su cuenta, este mensaje imita exactamente el diseño utilizado por la entidad para comunicarse con sus clientes.
- ❖ El mensaje puede incluir un formulario para evitar los datos solicitados, aunque lo más habitual es que incluya un enlace a una página donde actualizar la información del personal.
- ❖ Esta página es exactamente igual al original de la entidad (sencillo de falsificar) y su dirección *url* es parecida, incluso puede ser idéntica, aprovechando fallos en los navegadores.
- ❖ Si se rellenan y se envían los datos de la página, pueden utilizar su identidad para operar en Internet.



3.3. Instalación del Servidor para phishing

Lo primero que hemos tenido que hacer es descargarnos un servidor Apache para simular el ataque en nuestro ordenador, en este caso, hemos utilizado el *Tomcat 6.0* que es un servidor *Web* con soporte de *Servlets* (objetos que corren dentro del contexto de un contenedor de *Servlet*), es decir, deriva de otra anterior como *Applet* que se referirá a pequeños programas escritos en Java y que se ejecutan en el contexto del navegador *Web* y *JSP's*, además puede funcionar como servidor *Web* por sí mismo.

Otras de las características importantes son:

- ❖ Los usuarios disponen de libre acceso a su código fuente.
- ❖ Funciona en cualquier Sistema Operativo que disponga de una máquina virtual Java.

Tomcat se basa en una jerarquía de directorios mostrada en *Tabla 7* y para su instalación incluye:

Bin	Arranque, cierre y otros Scripts y ejecutables
Common	Clases comunes que puede utilizar Catalina que no es más que el nombre del contenedor de Servlets y las aplicaciones Web
Conf	Ficheros XML y los correspondientes DTD para la configuración del Tomcat
Logs	Logs de Catalina y de las aplicaciones
Server	Clases utilizadas solamente por Catalina.
Shared	Clases compartidas por todas las aplicaciones Webs
Webspps	Directorio que contiene las aplicaciones Webs
Work	Almacenamiento temporal de ficheros y directorios

Tabla 7. Jerarquía de directorios.

Antes de nada es importante señalar unos requisitos que hay que cumplir para escribir *Servlets*. Una vez instalado el contenedor, existirá un directorio "*webapps*", en él colocaremos todas las aplicaciones *Web* que hagamos.

Lo ideal es que dentro de este directorio nos creamos otro directorio y allí dentro, otra estructura de directorios que incluirán uno llamado "*WEB-INF*" y dentro de ese, otro con el nombre "*clases*".

En el directorio *ROOT* debería encontrarse la página *Web* principal.

Para nuestra práctica hemos hecho el directorio "*miServlet*" quedando de la siguiente estructura:



```
+ → Tomcat 6.0
|
+ → webapps
|
+ → miServlet
| error.html
|
+ → WEB-INF
| web.xml
|
+ → clases
  SimpleServlet.java
```

Figura 14. Directorio "miServlet".

Hemos tenido que modificar el `/WEB-INF/web.xml` → Este archivo contiene elementos de configuración del WAR como: Página de Inicio, Ubicación ("Mapeo") de *Servlets*, parámetros para componentes adicionales tales como "Struts" (es una herramienta de soporte para el desarrollo de aplicaciones Web) y otros elementos como manejo de errores.

```
<servlet>
  <servlet-name>SimpleServlet</servlet-name>
  <servlet-class>SimpleServlet</servlet-class>
</servlet>
  <servlet-mapping>
  <servlet-name>SimpleServlet</servlet-name>
  <url-pattern>/SimpleServlet</url-pattern>
  </servlet-mapping>
```

Figura 15. Archivo WEB-INF/web.xml.

3.4. Puesta en marcha del phishing

La recepción de un correo electrónico como se muestra en la *Figura 16*, de la "supuesta" pagina original que en este caso nos hemos basado en *PayPal* que consiste en un comercio electrónico en Internet que permite la transferencia de dinero entre usuarios que tengan email.



PayPal

Querido cliente de Paypal,

Recientemente hemos revisado su cuenta, y hemos descubierto una transacción dudosa en su cuenta.

Proteger su cuenta es nuestra prioridad en lo que a nosotros nos concierne. Como una medida preventiva tenemos limitado el acceso a su información.

Características de Paypal: no es necesario renovar tu cuenta, simplemente haciendo click en "**Centro de Resolución**" para confirmar su cuenta como miembro de Paypal.

- Su nombre de Paypal y su contraseña.
- Confirmar su identidad como miembro de Paypal.

Por favor confirma la información de su cuenta haciendo click en [Centro de Resolución](#) y completa "los pasos para quitar sus limitaciones".

Por favor no responda a este mensaje.

Copyright 1999-2008 Paypal. Todos los derechos reservados.

Figura 16. Recepción de un correo electrónico falso. Intento de Phishing.

El correo electrónico lo principal que tiene es vínculo hacia una página falsa, si el usuario hace click en el vínculo que en nuestro caso es [Centro de Resolución](#) nos enlazara con nuestra página principal llamada [index.html](#).

Para obtener una página similar a la original nos hemos basado en la opción que te da tanto Internet Explorer como Mozilla Firefox en *Archivo* → *Guardar como...* → *index.html* y lo único que tenemos que hacer es guardarla en el archivo principal del *Tomcat*, en este caso *ROOT*.

Para llamarla desde un navegador tendríamos que poner la siguiente URL:

<http://localhost:8080/index.html>

La página principal tendrá el siguiente aspecto como podemos observar en la *Figura 17*:



Figura 17. Supuesta página original.

Una vez estamos en la supuesta página el usuario, debería teclear su correo electrónico y su contraseña de *Paypal*, inmediatamente después el usuario verá otra página en la que muestra un mensaje de error, para obtener esta página de error hemos utilizado el mismo método que en la página principal pero en este caso la hemos guardado en *Tomcat 6.0\webapps\miServlet* para respetar los requisitos mencionados anteriormente.

Para enlazar la página principal con la página de error hemos tenido que cambiar el *link* de la página original de la siguiente manera:

```
<form class="rosetta" id="rosetta" method="post"
action="http://localhost:8080/miServlet/SimpleServlet">
```

Figura 18. Enlazar página principal con la de error.

De esta manera nos redirigirá a la página de error.



Para llamarla desde un navegador tendríamos que poner la siguiente *URL*:

<http://localhost:8080/miServlet/SimpleServlet>

La página de error tendrá el siguiente aspecto:



Figura 19. Página de error.

Una vez llegados a este punto mediante un archivo *.txt* hemos podido obtener y guardar tanto el *login* y *password*, como se observa en la Figura 21, para ello hemos tenido que implementar en el directorio:

```
\Tomcat 6.0\webapps\miServlet\WEB-INF\classes
```

, que es una clase java para obtener tanto el *login* como el *password*.

Nota: Es necesario cambiar unos campos de la página principal y llamarlos del mismo modo que en la clase Java, en nuestro caso son el *login* y *password*:



```
<input id="login" name="login" value="" type="text"></p>  
<input autocomplete="off" id="password" name="password" value=""  
  type="password"><script type="text/javascript">
```

Figura 20. Clase Java para obtener login y password.

El archivo tiene el siguiente aspecto:

```
Archivo Edición Formato Ver Ayuda  
login:sdd password:sdsd
```

Figura 21. Archivo *.txt.



CAPITULO 4. Escáneres de Redes

4.1. Introducción

El **escáner de red** (también denominado “analyzer de vulnerabilidades”) es una aplicación que permite realizar una verificación de seguridad en una red de forma automática mediante el análisis de los puertos abiertos en uno de los equipos o en toda la red. El proceso de análisis utiliza sondas (solicitudes) que permiten determinar los servicios que se están ejecutando en un *host* remoto.

Esta herramienta permite identificar los riesgos de seguridad. En general, con este tipo de herramienta es posible efectuar un análisis en una serie o lista de direcciones *IP* a fin de realizar una verificación completa de una red.

El funcionamiento del escáner de red permite identificar los puertos que están abiertos en un sistema al enviar solicitudes sucesivas a diversos puertos, además de analizar las respuestas para determinar cuáles están activos.

Mediante un análisis exhaustivo de la estructura de los paquetes *TCP/IP* recibidos, los escáneres de seguridad avanzados pueden identificar, a veces, qué sistema operativo ésta utilizando el equipo remoto, así como las versiones de las aplicaciones asociadas con los puertos y, cuando sea necesario, recomendar actualizaciones (esto se conoce como caracterización de la versión).

En general, se usan dos métodos:

- ❖ **Adquisición activa de información**: consiste en enviar una gran cantidad de paquetes con encabezados característicos que normalmente no cumplen con las recomendaciones y analizar las respuestas para identificar la versión utilizada, esto posibilita su diferenciación.
- ❖ **Adquisición pasiva de información** (también denominado análisis pasivo o análisis no agresivo): es un método mucho menos invasivo, que reduce la probabilidad de ser detectado por un sistema detector de intrusiones. Funciona de modo similar que el anterior y efectúa un análisis de los campos de datagramas *IP* que circulan en una red utilizando un rastreador de puertos. Dada su naturaleza pasiva, la versión analiza los cambios. Como todas ellas utilizan protocolos ligeramente diferentes, los valores de campo los divide en una serie de fragmentos, lo que requiere un tiempo de análisis mucho más prolongado. Por ello, este tipo de análisis es muy difícil e incluso imposible de detectar en determinadas ocasiones.



Los escáneres de seguridad son herramientas sumamente útiles para los administradores de sistemas y redes, ya que les permite supervisar la seguridad de todos los equipos que están a su cargo.

Sin embargo, esta herramienta puede ser utilizada por los piratas informáticos para identificar las vulnerabilidades del sistema.

Nosotros trabajaremos con *Nmap* y *Nessus* que son dos ejemplos de este tipo de herramientas. *Nmap* (“mapeador de redes”) es un rastreador de puertos diseñado para explorar grandes redes y determinar qué equipos están activos y cuáles son los servicios *TCP* y *UDP* que ofrecen. *Nmap* es un programa de código abierto, disponible en <http://www.insecure.org/nmap>. En esta dirección, también podemos encontrar el paquete *Nmapfe*, una interfaz gráfica para *Nmap*.

Nessus es un programa de escaneo de vulnerabilidades en diversos sistemas operativos (se puede descargar desde <http://nessus.org>). Está diseñado siguiendo una arquitectura cliente/servidor. Consiste en el servidor/demonio, *nessusd*, que se encarga de realizar la búsqueda de vulnerabilidades en los sistemas remotos (lanza ataques). Mientras que el cliente *Nessus* proporciona al usuario una atractiva interfaz *X11/GTK+*, que permite conectarse al servidor, configurar y lanzar los rastreos. Una vez inspeccionado un equipo o una red, *Nessus* elabora un informe exhaustivo en el que se describen los riesgos encontrados, y posibles soluciones, junto con una referencia *CVE* (*Common Vulnerabilities and Exposures*). *CVE* es una enorme base de datos, disponible en <http://cve.mitre.org>, donde se pueden encontrar información sobre problemas de seguridad conocidos.

Los ataques reciben el nombre de “*plugins*”, un concepto realmente importante detrás de *Nessus*, es sin lugar a dudas el de la existencia de un lenguaje de *scripting* excepcional, conocido como *Nenta* ampliamente extensible. De este modo, *Nessus* no solo crece en cuanto a la cantidad de chequeos que realiza una función de los *plugins* que desarrolle en equipo interno, sino que a su vez, se nutre de aquellos desarrollados por la comunidad.

Como ventaja podemos decir que el *ASL* (*Nessus Attack Scripting Language*) está específicamente diseñado a efectos de que los analistas de seguridad tengan la posibilidad de crear en forma rápida y sencilla sus propios *plugins* de chequeo de vulnerabilidades, haciendo de *Nessus* una herramienta adicional, no está de más agregar que precisamente, la existencia de *NASL* hace de *Nessus* una herramienta imprescindible, para aquellos que deseen crear sus propio *set* de pruebas específicas respecto de aquellos protocolos y servicios que eventualmente podrían ser por que no, particulares para sus propias redes y sistemas.

Existen varias familias de *plugins*: puertas traseras, denegación de servicio o acceso *root* remoto. En la página *WEB* de *Nessus* se puede encontrar información sobre ellos. La librería de *plugins* no es algo estático, sino que se incrementa con la aparición y conocimiento de nuevas vulnerabilidades. Se pueden programar *plugins* utilizando lenguajes *C*, o el lenguaje *NASL* (*Nessus Attack Scripting Language*).



4.2. Nmap

4.2.1. Características Nmap

Las principales características de *Nmap* se usan para:

- ❖ Descubrimiento de servidores: Identificación de ordenadores en una red.
- ❖ Identifica puertos abiertos en un ordenador objetivo.
- ❖ Determina los servicios está ejecutando en un ordenador.
- ❖ Determina el sistema operativo y que versión utiliza dicho ordenador.
- ❖ Obtiene ciertas características del hardware del ordenador objetivo.

4.2.2. Funcionamiento Nmap

Para ejecutar cualquiera de los comandos que vamos a utilizar es necesario tener permisos de supervisor, por lo que se recomienda realizar la práctica como usuario *root*.

Los pasos que hemos realizado para el funcionamiento de Nmap, son los siguientes:

1. Descargamos el programa.
2. Instalamos el programa.
3. Ejecutar.
4. Identificar la IP de nuestra red.
5. Escaneo de la red del laboratorio.

Lo primero que tenemos que realizar es la instalación del programa Nmap para ello ejecutamos el comando:

```
# apt-get install nmap
# apt-get nmapfe
```

En el directorio donde se encuentran estos dos programas podemos saberlo mediante el comando que nos proporciona Linux con la instrucción *whereis*:

```
# /usr/bin/nmap
# /usr/bin/nmapfe
```

Realizamos un rastreo de puertos en toda la red del laboratorio, para ello tenemos que ejecutar el programa *Nmapfe*, el cual nos llevará a la realización de un *scan* del laboratorio como se puede ver en la *Figura 22*:

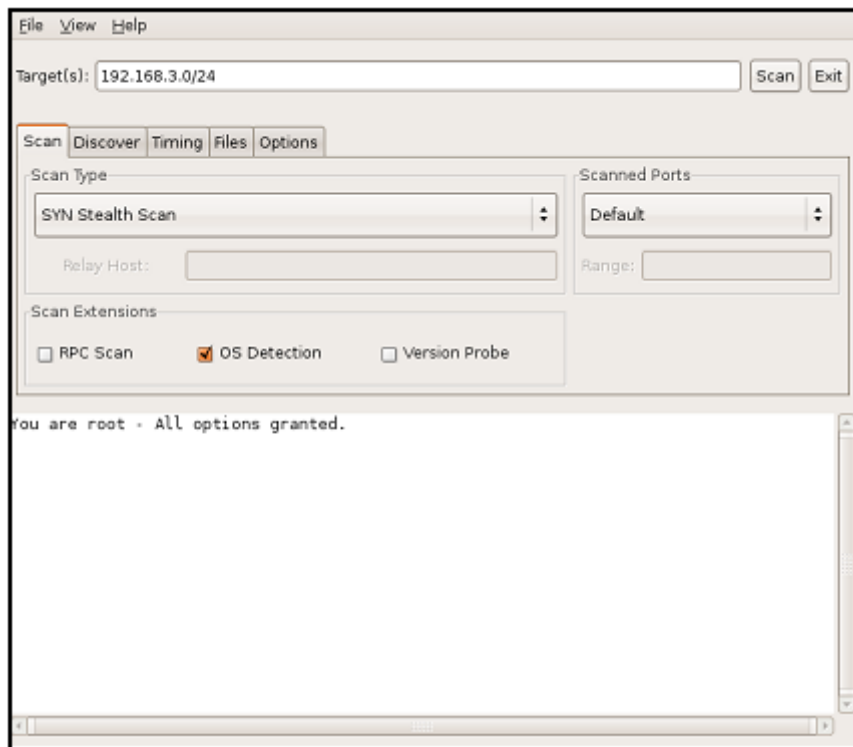


Figura 22. Scan del laboratorio.

Informe de los Pc del laboratorio:

```
Starting Nmap 4.53 ( http://insecure.org ) at 2008-09-10 18:09 CEST
Interesting ports on 192.168.3.101:
Not shown: 1714 closed ports
PORT STATE SERVICE
23/tcp open telnet
MAC Address: 00:0A:F4:DB:2A:41 (Cisco Systems)
Device type: router
Running: Cisco IOS 12.X
OS details: Cisco 806, 1712, 1721, or 2600 router (IOS 12.2 - 12.3)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=265 (Good luck!)
IP ID Sequence Generation: All zeros

Host 192.168.3.253 appears to be up ... good.
Interesting ports on 192.168.3.253:
Not shown: 1712 closed ports
PORT STATE SERVICE
23/tcp open telnet
80/tcp open http
1024/tcp open kdm
MAC Address: 00:30:F1:55:01:22 (Accton Technology)

Device type: switch
```



```
Running: 3Com embedded
OS details: 3Com SuperStack 3 Switch 4300
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=17 (Good luck!)
IP ID Sequence Generation: Incremental
```

```
Host 192.168.3.254 appears to be up ... good.
Interesting ports on 192.168.3.254:
Not shown: 1702 closed ports
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
80/tcp open http
111/tcp open rpcbind
139/tcp open netbios-ssn
443/tcp open https
445/tcp open microsoft-ds
710/tcp open unknown
831/tcp open unknown
882/tcp open unknown
1241/tcp open nessus
2049/tcp open nfs
32770/tcp open sometimes-rpc3
MAC Address: 00:0B:CD:08:DC:D0 (Hewlett Packard)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.20
Uptime: 15.064 days (since Mon Aug 25 19:00:21 2008)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=201 (Good luck!)
IP ID Sequence Generation: All zeros
```

```
Read data files from: /usr/share/nmap
OS detection performed. Please report any incorrect results at
http://nmap.org/submit/.
Nmap done: 256 IP addresses (13 hosts up) scanned in 97.815 seconds
Raw packets sent: 24330 (1.094MB) | Rcvd: 24842 (1.146MB)
```

Como podemos observar se han rastreado **13 host**.

Ahora para proceder a la desinstalación de estos dos paquetes introducimos los siguientes comandos:

```
# apt-get remove nmap
# apt-get remove nmapfe
```



4.3. Nessus

En *Nessus*, la arquitectura de la aplicación es la que se observa en la *Figura 23*.

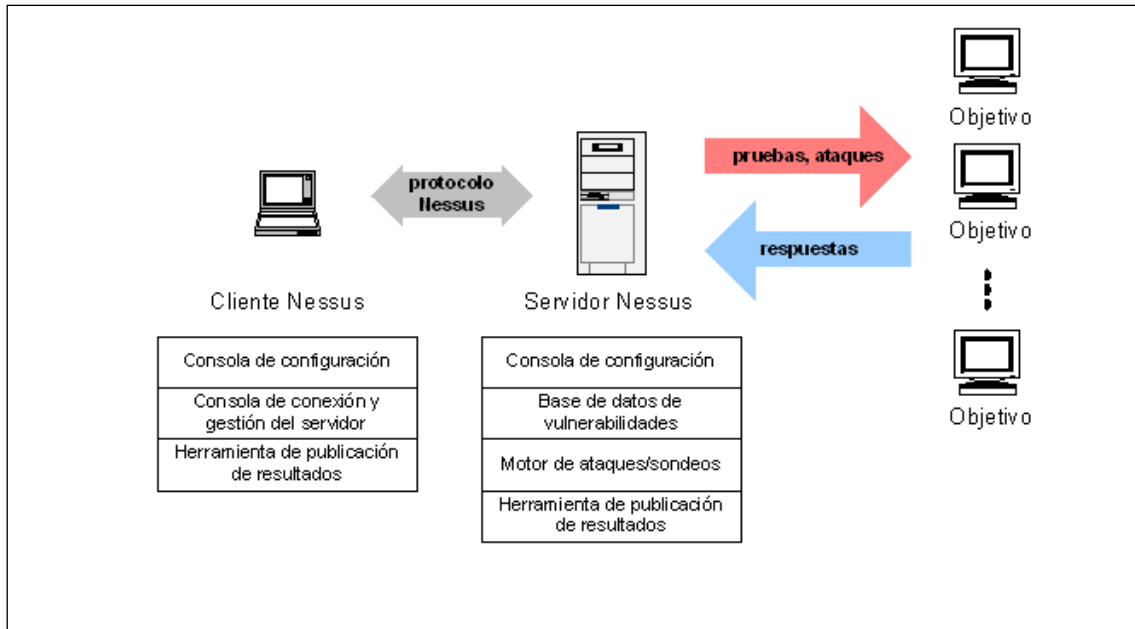


Figura 23. Arquitectura Nessus.

El cliente nessus solicita al servidor *nessusd* que lance una serie de ataques sobre una serie de equipos o redes objetivo (“*target*”). Aunque en la figura hay tres partes diferenciales:

- ❖ Cliente.
- ❖ Servidor.
- ❖ *Target*.

Tras la instalación de los dos programas *Nessus* y *Nessusd* comprobamos donde se encuentra su ubicación y la existencia de los mismos, (por ejemplo, usando el comando *whereis*):

```
# /usr/bin/nessus
# /usr/bin/nessusd
```

Si todo se ha instalado correctamente, *Nessus* habrá sido descargado, instalado y se encontrará listo para ser configurado, ya que, al ser ejecutado por primera vez, *Nessus* requerirá al menos un par de ajustes antes de poder ser utilizado.

Para empezar, debemos crear un usuario válido con las siguientes características: (1) el usuario sólo podrá acceder al servidor desde *PC* situados en el laboratorio *IT-3*, (2) el usuario sólo podrá lanzar ataques a *PC* situados en el laboratorio *IT-3*, a excepción del servidor (192.168.3.59) y (3) el tipo de autenticación será “*pass*”, cuyo usuario es utilizado con *nessusd*.



```
# nessus-adduser
```

Using /var/tmp as a temporary file holder

Add a new nessusd user

```
-----  
Login: sito  
Authentication (pass/cert) [pass]:  
Login password:  
Login password (again):
```

User rules

```
-----  
nessusd has a rules system which allows you to restrict the hosts  
that sito has the right to test. For instance, you may want  
him to be able to scan his own host only.
```

Please see the `nessus-adduser(8)` man page for the rules syntax

Enter the rules for this user, and hit `ctrl-D` once you are done :
(the user can have an empty rules set)

```
deny 192.168.3.59  
accept 192.168.3.0/24  
default deny
```

ctrl+d

```
Login : sito  
Password : *****  
DN :  
Rules :  
deny 192.168.3.59  
accept 192.168.3.0/24  
default deny
```

```
Is that ok ? (y/n) [y]  
user added.
```

Ahora que tenemos nuestro usuario, vamos a crear un certificado para el servidor `nessusd`.
Esto se consigue ejecutando el comando:

```
# nessus-mkcert
```



Creation of the Nessus SSL Certificate

This script will now ask you the relevant information to create the SSLcertificate of Nessus. Note that this information will **NOT** be sent to anybody (everything stays local), but anyone with the ability to connect to your Nessus daemon will be able to retrieve this information.

```
CA certificate life time in days [1460]:
Server certificate life time in days [365]:
Your country (two letter code) [FR]: ES
Your state or province name [none]: Murcia
Your location (e.g. town) [Paris]: Cartagena
Your organization [Nessus Users United]:
```

Creation of the Nessus SSL Certificate

Congratulations. Your server certificate was properly created.

/etc/nessus/nessusd.conf updated

The following files were created :

```
. Certification authority :
Certificate = /usr/com/nessus/CA/cacert.pem
Private key = /var/lib/nessus/CA/cakey.pem

. Nessus Server :
Certificate = /usr/com/nessus/CA/servercert.pem
Private key = /var/lib/nessus/CA/serverkey.pem
```

Press [ENTER] to exit

Tenemos nuestro usuario y certificado, quizás ahora queramos revisar las opciones de configuración.

En nuestro caso el archivo de configuración del demonio *Nessus* (*Nessusd*), para lo cual podrás editar el fichero que se encuentra por defecto en */etc/nessus/nessusd.conf*, aunque en verdad las opciones configuradas por defecto, probablemente sirvan a los efectos de llevar a cabo estas pruebas, nosotros lo modificaremos con las siguientes características:

- ❖ Número máximo de *PC* que se pueden inspeccionar a la vez sean 10.
- ❖ El número de *Plugins* lanzados de forma simultánea contra un *PC*, sea 5.



```
# /etc/nessus/nessusd.conf
```

Los parámetros que tenemos que modificar son los siguientes:

```
# Maximum number of simultaneous hosts tested :
max_hosts = 30 -> max_host = 10

# Maximum number of simultaneous checks against each host tested :
max_checks = 10 -> max_checks = 5

usage : nessusd [-vcphdLDCR] [-a address] [ -S <ip[,ip,...]> ]

a <address> : listen on <address>
S <ip[,ip,...]>: send packets with a source IP of <ip[,ip...]>
v : shows version number
h : shows this help
p <number> : use port number <number>
c <filename> : alternate configuration file to use
(default : /etc/nessus/nessusd.conf)
D : runs in daemon mode
d : dumps the nessusd compilation options
q : quiet (do not issue any message to stdout)
```

Ahora lanzamos el demonio nessusd en modo *background*, para lo cual bastará con ejecutar el comando mostrado a continuación:

```
# nessusd -D
```

```
Loading the Nessus plugins...
```

```
-----
You are running a version of Nessus which is not configured to receive
a full plugin feed. As a result, your security audits will produce
incomplete results.
```

```
To obtain a complete plugin feed, you need to register your Nessus
scanner at the following URL :
```

```
http://www.nessus.org/register/
```

```
-----
Loading the plugins... 153 (out of 1122)smb_func.inc: No such file or
directory
```

```
Loading the plugins... 663 (out of 1122)snmp_func.inc: No such file or
directory
```




```
byte_func.inc: No such file or directory
Loading the plugins... 918 (out of 1122) smb_func.inc: No such file or
directory
All plugins loaded
```

```
Ir a la pestaña Target
Target(s) : 192.168.3.0/24
```

Nota: El servidor *Nessusd* tiene que estar en todo momento corriendo para arrancar el cliente *Nessus*.

Para ello realizamos los pasos necesarios a los efectos de lanzar un sencillo escaneo de vulnerabilidades contra alguno de nuestros equipos de prueba dispuestos a tal fin.

Para la actualización de los *plugins* como hemos mencionado, *nessusd* es un motor específicamente diseñado con el propósito de ejecutar de forma ordenada y eficiente una serie de chequeos de seguridad dispuestos en forma de *plugins*. Debido a ello, es de especial importancia actualizar la base de datos de *plugins* de nuestra instalación *Nessus*, antes de cada ejecución. De esta forma, podemos asegurarnos el contar con los últimos chequeos de vulnerabilidades disponibles. A tal efecto, *Nessus* nos provee de un comando específicamente diseñado con el objeto de realizar esta función:

```
# nessus-update-plugins
```

Nota: Las últimas versiones de *Nessus*, exigen que el usuario realice un sencillo registro *on-line* a través de la *web*, a efectos de obtener y registrar en su instalación local de *Nessus*, el código requerido a la hora de realizar la actualización de la totalidad de los *plugins* disponibles se encuentra en:

<http://www.nessus.org/plugins/index.php?view=register>

```
# nessus-fetch --register E4E3-887A-4C6A-E455-7551
```

```
Your activation code has been registered properly - thank you.
Now fetching the newest plugin set from plugins.nessus.org...
Your Nessus installation is now up-to-date.
Make sure to call regularly use the command 'nessus-update-plugins' to
stay up-to-date To automate the update process, please visit
http://www.nessus.org/documentation/index.php?doc=cron
```

Vemos después de la actualización de cuantos *plugins* tenemos disponibles:

```
# nessusd
```



Loading the plugins... 20604 (out of 23526)
All plugins loaded

Habiendo cargado los últimos *plugins* disponibles, volveremos a ejecutar el cliente *Nessus*, tal como fuera mencionado con anterioridad (*nessusd / nessus*), para que luego de identificarnos con las credenciales correctas podamos avanzar con nuestro escaneo de prueba.

```
# nessusd
```

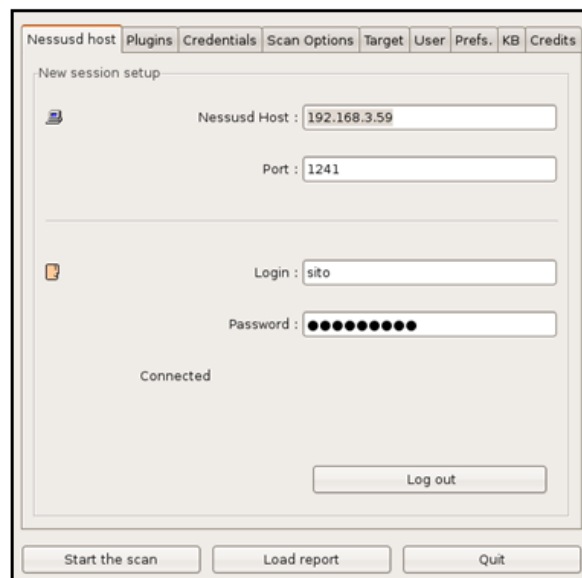


Figura 24. Ejecución cliente *Nessus*.

A partir de estar logueados a través de nuestro cliente con el motor *nessusd*, muchas serán las opciones a nuestra disposición a la hora de configurar las opciones a utilizar en nuestro primero escaneo de vulnerabilidades.

Por ultimo, y teniendo en cuenta que nuestro objetivo es tan solo mostrar los pasos mínimos y necesarios a los efectos de poner a funcionar nuestro primer escaneo, bastará con acceder a la pestaña "*Target*", escribir la IP o nombre de host del objetivo y presionar el botón "*Start the Scan*", para que *Nessus* comience a realizar su trabajo.

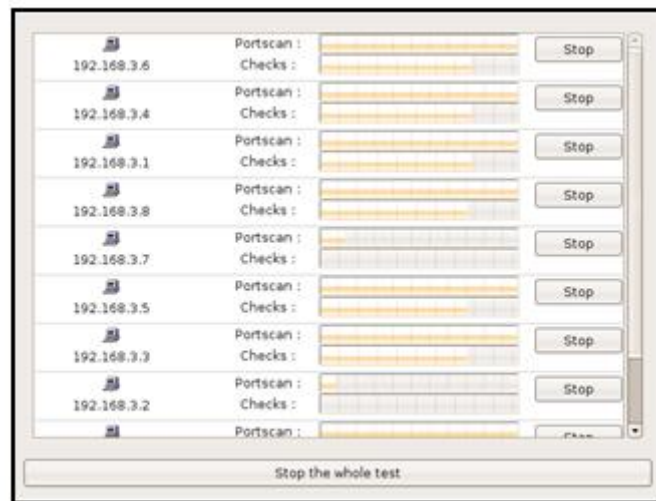


Figura 25. Inicio del escaneo.

Al terminar el proceso de chequeos, se abrirá una nueva ventana, *Figura 26* que contiene el resultado del escaneo de vulnerabilidades, el cual mostrará cada uno:

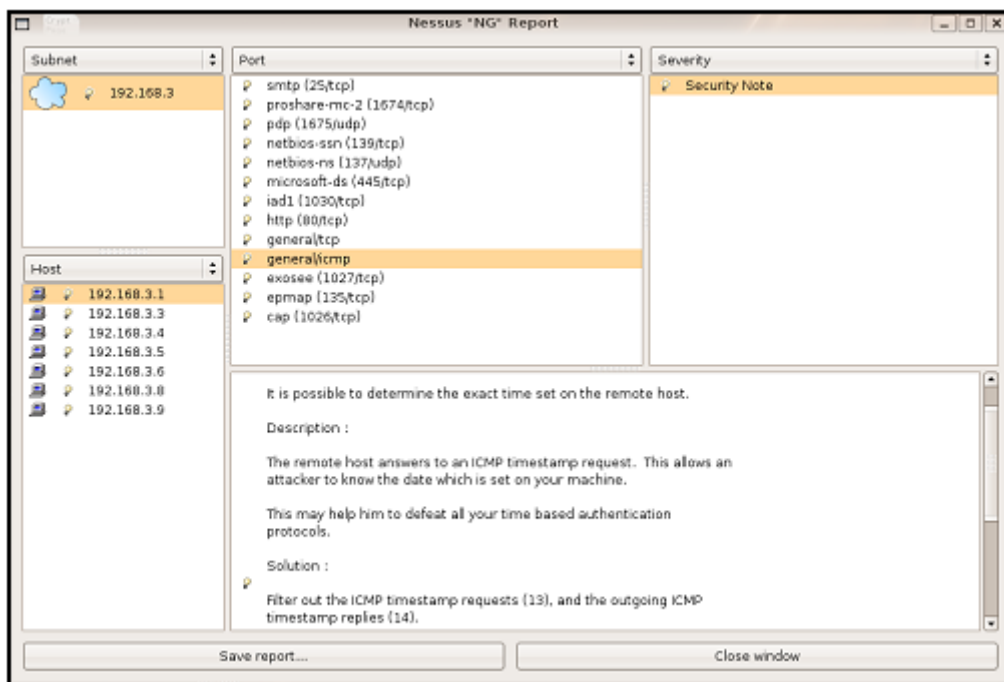


Figura 26. Proceso escaneo de vulnerabilidades.



Nessus Report

The Nessus Security Scanner was used to assess the security of 7 hosts
0 security warning has been found
147 security notes have been found

Part I: Graphical Summary:

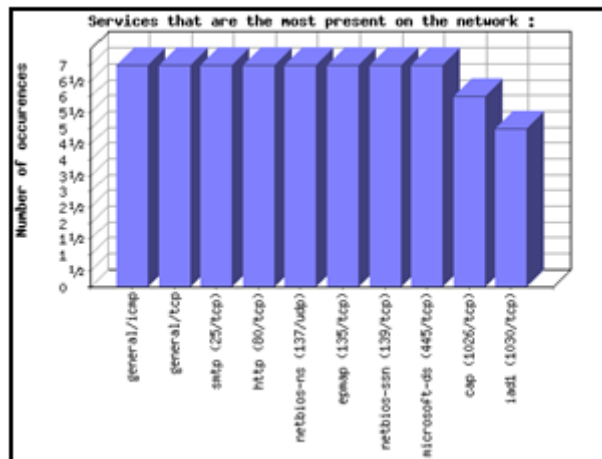


Figura 27. Resultado de Nessus.

Part II. Results, by host:

192.168.3.1 (found 21 security notes)
192.168.3.3 (found 21 security notes)
192.168.3.4 (found 21 security notes)
192.168.3.5 (found 21 security notes)
192.168.3.6 (found 21 security notes)
192.168.3.8 (found 21 security notes)
192.168.3.9 (found 21 security notes)

Tabla 8. Hosts.



Para la máquina 192.168.3.1 nos muestra el siguiente resultado:

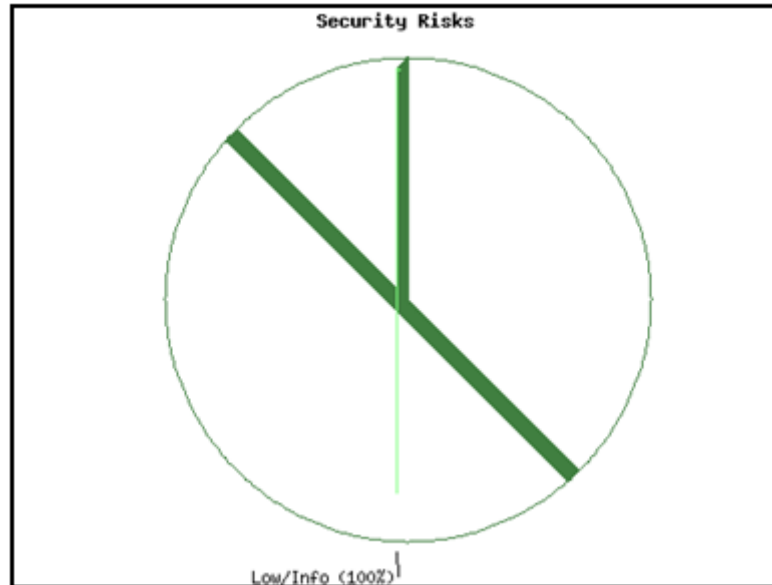


Figura 28. Resultado de la máquina.

Para terminar nuestro escaneo de redes vamos a desinstalar los programas, para ello ejecutamos el comando:

```
# apt-get remove nessus  
# apt-get remove nessusd
```



CAPITULO 5. Autenticación. Radius y EAP

5.1. Introducción

5.1.1. Radius (Remote Authentication Dial – In Users Service)

Es un protocolo de autenticación de usuario remoto, la autorización y la contabilidad.

RADIUS permite la gestión centralizada de datos de autenticación, como nombres de usuario y contraseñas. Cuando un usuario intenta acceder a un cliente *RADIUS*, como un router, el router envía la solicitud de autenticación al servidor *RADIUS*. La comunicación entre el cliente *RADIUS* y el servidor *RADIUS* son autenticados mediante la utilización de un secreto compartido, que no se transmite a través de la red. Desarrollado inicialmente para el acceso remoto *dial-up* (marcado manual). *RADIUS* es soportado actualmente por servidores *VPN* (*Virtual Private Network*), puntos de accesos inalámbricos, conmutadores de autenticación *Ethernet*, acceso *DSL* (*Digital Subscriber Line*) y otros tipos de redes de acceso.

El protocolo *RADIUS* se describe en la RFC 2865 (www.ietf.org).

El servidor *RADIUS* comprueba las credenciales del cliente, indicando mediante un mensaje de respuesta si se autoriza o no la petición del cliente *RADIUS*. El cliente *RADIUS* también puede enviar al servidor *RADIUS* mediante mensajes de contabilidad (*Accounting*). El servidor *RADIUS* puede almacenar los datos de autenticación a nivel local, sino que también puede almacenar los datos de autenticación en una base de datos *SQL* externa o *Unix*, en el archivo */etc/passwd*. El servidor *RADIUS* también puede conectar en un *PAM* (*Pluggable Servicio de autenticación*) para recuperar la arquitectura de datos. Adicionalmente, el estándar *RADIUS* soporta el uso de servidores *Proxy RADIUS*. Un *proxy RADIUS* es un equipo que remite mensajes *RADIUS* entre el cliente *RADIUS*, servidores *RADIUS* u otros proxies *RADIUS*.

Los mensajes *RADIUS* nunca se envían entre el cliente de acceso y el servidor de acceso.

Los mensajes *RADIUS* se envían como mensajes de Protocolo de datagramas de usuario *UDP* (*User Datagram Protocol*). El puerto *UDP* 1812 se utiliza para los mensajes de autenticación *RADIUS* y el 1813 para los mensajes de administración de cuentas *RADIUS*. Puede que algunos servidores de acceso a la red utilicen el puerto *UDP* 1645 para los mensajes de autenticación *RADIUS* y el 1646 para los mensajes de administración de cuentas *RADIUS*. De manera predeterminada, *IAS* admite la recepción de mensajes *RADIUS* destinados a ambos grupos de puertos *UDP*. La carga *UDP* de un paquete *RADIUS* sólo incluye un mensaje *RADIUS*.



En los documentos *RFC 2865* y *2866* se definen los siguientes tipos de mensajes *RADIUS*:

- ❖ **Access – Request** (solicitud de acceso): Enviado por un cliente *RADIUS* para solicitar autenticación de un intento de conexión.
- ❖ **Access – Accept** (aceptación de acceso): Enviado por un servidor *RADIUS* como respuesta a un mensaje *Access-Request*. En él se informa al cliente *RADIUS* de que se ha autenticado y autorizado el intento de conexión.
- ❖ **Access – Reject** (rechazo de acceso): Enviado por un servidor *RADIUS* como respuesta a un mensaje *Access-Request*. En él se informa al cliente *RADIUS* de que se ha rechazado el intento de conexión. Un servidor *RADIUS* envía este mensaje si las credenciales no son auténticas o si no se ha autorizado el intento de conexión.
- ❖ **Access – Challenge** (desafío de acceso): Enviado por un servidor *RADIUS* como respuesta a un mensaje *Access-Request*. Este mensaje es un desafío al cliente *RADIUS* que exige una respuesta.
- ❖ **Accounting – Request** (solicitud de administración de cuentas): Enviado por un cliente *RADIUS* para especificar información de administración de cuentas de una conexión que se ha aceptado.
- ❖ **Accounting – Response** (respuesta de administración de cuentas): Enviado por el servidor *RADIUS* como respuesta a un mensaje de solicitud de administración de cuentas. En este mensaje se confirman la recepción y el procesamiento correctos del mensaje de solicitud de administración de cuentas.

Un mensaje *RADIUS* está formado por un encabezado *RADIUS* y cero o más atributos *RADIUS*. Cada atributo *RADIUS* especifica una información determinada acerca del intento de conexión.

5.1.2. EAPOL (Extensible Authentication Protocol over LAN)

Para llegar a comprender qué es el estándar *IEEE 802.1X* es necesario conocer además dos conceptos: *PPP* y *EAP*. Estas son las bases de la nueva técnica de seguridad para *LAN* inalámbricas.

El *Point-to-Point* es el más popular, siendo uno de los protocolos más utilizados en los sistemas de acceso a Internet por marcación, así como por algunos *ISP* para autenticación por cable módem y *DSL*, en la modalidad de *PPP* sobre Ethernet.

Una de las principales preocupaciones de la mayoría de las empresas es hacer que sus sistemas de seguridad para acceder a los recursos de la compañía sean algo más que teclear el nombre de un usuario y una contraseña, por lo que se ha diseñado un nuevo protocolo de autenticación denominado *EAP (Extensible Authentication Protocol)*. *EAP* se encuentra dentro del protocolo de autenticación *PPP* y proporciona un marco general compatible con diversos métodos de autenticación. El nuevo *EAP* está diseñado para disuadir a los usuarios de la



implementación de sistemas de autenticación propietarios y permitir desde las contraseñas hasta los certificados de clave pública.

Con el estándar *EAP* la interoperabilidad y la compatibilidad de los métodos de autenticación llegará a ser de lo más simple y solamente es necesario que el usuario y el servidor de autenticación estén coordinados. Al soportar autenticación *EAP*, el servidor *RAS* se libera de actuar de intermediario, y sólo se ocupa de empaquetar y re empaquetar los paquetes *EAP* a un servidor *RADIUS*, que se encargará de la autenticación real.

Todo lo dicho anteriormente nos lleva al estándar *IEEE 802.1X*, que, simplemente, se trata de una norma para pasar *EAP* por una *LAN* cableada o inalámbrica. Con *802.1X*, el usuario empaqueta los mensajes *EAP* en tramas *Ethernet* sin tener que recurrir a *PPP*. Se trata de autenticación y nada más. En la definición del **concepto 802.1X** es necesario llegar a conocer tres términos:

- ❖ **Suplicante o cliente:** la máquina que solicita el acceso a la red.
- ❖ **Autenticador:** *Software* con el único propósito de autorizar a un determinado cliente.
- ❖ **Servidor de autenticación:** un servidor *RADIUS* que proporciona servicios de autenticación al autenticador. Se comunica con el suplicante haciendo uso del protocolo de encapsulación *EAPOL*.
- ❖ **PAE (*Port Access Entity*):** Entidad software asociada con cada puerto que soporta funcionalidad tanto de cliente como de autenticador.
- ❖ **Puerto controlado:** cualquier puerto del *switch* que tenga activada la característica de seguridad basada en *EAP*.

5.1.2.1. Modo de operación

El **protocolo 802.1X** se denomina *EAP (encapsulation over LAN)* y actualmente está definido para redes *LAN* del tipo *Ethernet*, cableadas e inalámbricas *802.11*, *Token-Ring* y *FDDI*.

EAPOL no es particularmente sofisticado, y, si bien ofrecen diversos modos de operación o funcionamiento del puerto controlado.

Dicho estado puede variar entre dos opciones:

- ❖ **Reenvío o *Forwarding*:** estado en el que se permite el paso de los paquetes a través del conmutador mediante el puerto en cuestión.
- ❖ **Bloqueo o cerrado:** estado en el que no se permite el paso de paquetes a través de dicho puerto.

El caso más común podría ser el que sigue y como podemos ver en la *Figura 29*:



1. El autenticador envía un paquete "EAP-Request/Identity" al suplicante en cuanto detecte que la conexión esté activa.
2. El suplicante envía un paquete "EAP-Response/Identity" al autenticador, que lo pasa al servidor de autenticación (RADIUS).
3. El servidor de autenticación pide la identificación al autenticador, como un sistema de contraseñas *token*. El autenticador lo desempaqueta desde IP y lo empaqueta de nuevo en EAPOL, y a continuación lo envía al suplicante. Según el método de autenticación variará la naturaleza y número de estos mensajes. Aunque EAP soporta autenticación basada sólo en cliente y autenticación mutua, esta última es la más apropiada para redes inalámbricas.
4. El suplicante responde a la petición de identificación vía el autenticador y traslada la respuesta al servidor de la autenticación.
5. Si el demandante proporciona la identidad apropiada, el servidor de autenticación responde con un mensaje de éxito, el cual se pasa al suplicante. Es en este momento cuando el identificador permite el acceso a la LAN, posiblemente con algunas restricciones en función de los atributos que devuelve el servidor de autenticación.

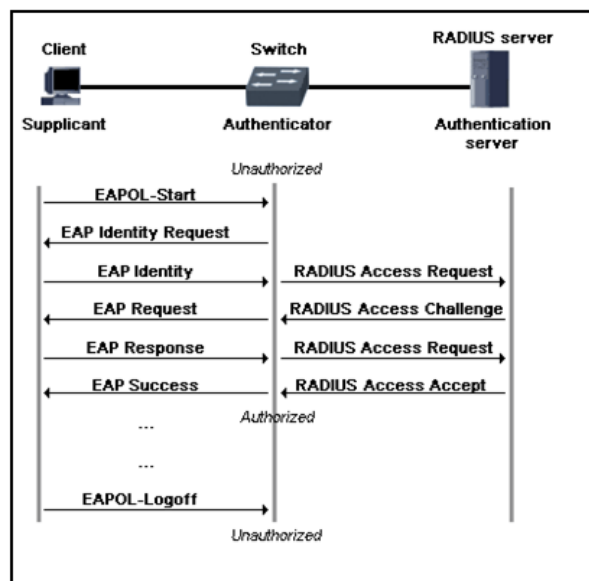


Figura 29. Modo de operación.

5.2. Desarrollo de las instalaciones

5.2.1. Instalación del servidor Radius

En primer lugar descargamos el archivo *FreeRadius.tar.gz* desde la web oficial del *FreeRadius* www.freeradius.org de forma gratuita.



Una vez descargado procedemos a instalar el software:

```
# sudo su
# mv /home/sito/Escritorio/freeradius.tar
# tar xzf freeradius.tar
# cd freeradius
# ./configure --prefix =/usr/local/radius
# make && make install
```

Una vez finalizado el proceso de instalación es necesario modificar algunas opciones de los archivos de configuración. Estos archivos de configuración están en el directorio: **/usr/local/radius/etc/raddb**

Los archivos de configuración más importantes son:

- ❖ **Clients.conf:** en este archivo se guarda un listado de las máquinas que pueden actuar como clientes del servidor *RADIUS*.
- ❖ **Users.conf:** en este archivo se almacenan una serie de entradas, cada una de las cuales se corresponde a un usuario, y donde se especifican una serie de parámetros para cada uno de ellos.
- ❖ **Radius.conf:** en el se configuran las opciones del propio servidor *RADIUS*, tales como el método de cifrado.

El archivo que modificamos es el **users**. En dicho archivo se incluyen todos los usuarios que pueden acceder al servidor *RADIUS* (a excepción de los usuarios con cuentas locales en la máquina donde se ejecuta el servidor), pudiendo especificarse multitud de parámetros para cada uno de ellos (tipo de autenticación, tipo de servicio, clave, etc....).

Configuración del archivo **users.conf** con las siguientes entradas:

```
alum1 Auth-type:=EAP, Cleartext-Password := "alum1pass"
      Service-Type = Administrative-User

alum2 Auth-type:=EAP, Cleartext-Password := "alum2pass"
      Service-Type = Administrative-User
```

Donde “Auth-type” indica el tipo de autenticación para un determinado usuario. Además del protocolo EAP otras posibles opciones son: System y Local.

La configuración del archivo **users** todavía no ha concluido. Falta proteger la interfaz de configuración y el acceso por telnet al *switch* mediante la autenticación de un identificador de usuario y su correspondiente clave almacenados también en el servidor *RADIUS*. Para ello deberá añadir un nuevo usuario a la lista de usuarios del fichero *users*, con privilegio



administrativo que puede acceder al interfaz web de configuración del *switch*. Añadimos para ello la siguiente entrada:

```
admin Auth-type:=EAP, Cleartext-Password := "administradorpass"  
Service-Type = Administrative-User
```

Configuración del archivo *clients.conf* con la siguiente entrada:

```
client 192.168.3.240 {  
    secret = alumno  
    shortname = Swith  
}
```

Ponemos en marcha el servidor *RADIUS*. Que se encuentra dentro del directorio */usr/local/radius/sbin/radiusd -X*.

5.2.2. Configuración del suplicante (Cliente EAP)

Descargamos el software *Odyssey Client* para Windows de la web:

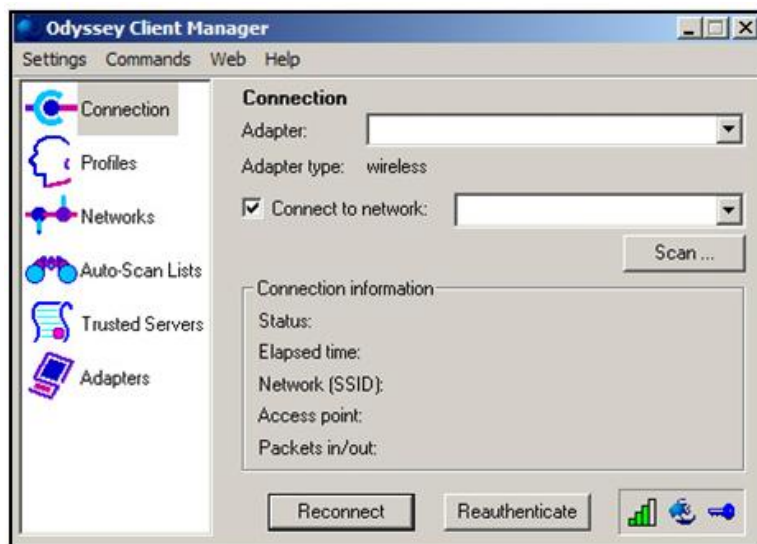


Figura 30. Odyssey Client.

Una vez instalado *Odyssey*, procedemos a su configuración:

1. Arrancamos el *Client Manager* (que permite configurar un cliente EAP).
2. Desactivamos el cliente con la opción *Settings and Disable Odyssey*.



3. En *Profiles*, se configuran los perfiles del usuario. Pulse el botón *Add*. En la pestaña *UserInfo* introduzca el nombre de usuario (alum1) y como nombre de perfil "Alumno1". Active las casillas *Permit login using y prom. for password*.
4. En la pestaña *Authentication* eliminamos todos los protocolos de autenticación de la misma. Pulse *Add* para añadir uno nuevo. Seleccionamos *EAP/MD5 Challenge*. Pulsamos OK.
5. Creamos un nuevo perfil para el usuario alum2.
6. Vamos a la opción *Networks*, donde se configuran las redes para las que está configurado el cliente EAP. Seleccionamos *<any>* y pulsamos *Properties*. Activamos las casillas *Connect to any available network y Autenticate using profile alum1*. Pulsamos OK.
7. En la opción *Adapters*, comprobamos que el adaptador de red del equipo está lista.
8. En la opción *Connection* elegimos el adaptador de red marcándolo en el menú desplegable de *Adapter*. Activamos la casilla *Connect using Profile alum1*.

5.2.3. Configuración del Autenticador (Switch Cisco Catalyst 2960)

Creamos una nueva conexión. Asignamos el puerto serie correspondiente (COM1 o COM2). Establecemos las siguientes propiedades:

Bits por segundo	9600
Bits de datos	8
Paridad	Par
Bits de parada	1
Control de flujo	Xon/Xoff

Tabla 9. Configuración del Autenticador.

Inicialmente el *switch* aparece en modo usuario (indicado por el símbolo >). Es el modo privilegiado el que da acceso a todos los comandos del *switch*. Para pasar a modo privilegiado (viene indicado por el símbolo #) ejecutamos el comando *enable*.



Para eliminar cualquier configuración anterior del *switch* ejecutamos los siguientes comandos:

```
Switch#delete flash:vlan.dat  
Switch#erase startup-config  
Switch#reload
```

Para asignar una dirección IP al *switch* ejecutamos los siguientes comandos:

```
Switch#configure terminal  
Switch(config)#interface VLAN1  
Switch(config-if)#ip address 192.168.3.240 255.255.255.0
```

Seguridad EAP en el switch:

Salimos del modo de configuración de interfaz con el comando ***exit***:

```
Switch(config)#
```

Habilitamos AAA y el método de autenticación:

```
Switch(config)#aaa new-model  
Switch#aaa authentication dot1x default group radius
```

Activamos la autenticación 802.1x en los interfaces correspondientes:

```
Switch(config)#interface fastethernet 0/1  
Switch(config-if)#dot1x port-control auto  
Switch(config-if)#switchport mode access  
Switch(config-if)#end  
Switch(config)#dot1x system-auth-control  
Switch#copy running-config startup-config
```

Configuramos del *switch* como cliente del servidor RADIUS, para que el *switch* pueda actuar como cliente del servidor RADIUS ejecutamos los siguientes comandos:

```
Switch#configure terminal  
Switch(config)#radius-server-host 192.168.3.59 authport 1645 key  
alumno  
Switch(config)#end  
Switch#copy running-config startup-config
```



Una vez configurado todo, el *switch* quedará configurado de la siguiente manera:

```
Current configuration : 1478 bytes
!
version 12.2
no service pad
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Switch
!
!
aaa new-model
aaa authentication dot1x default group radius
!
aaa session-id common
ip subnet-zero
!
!
!
!
dot1x system-auth-control
no file verify auto
spanning-tree mode pvst
spanning-tree extend system-id
!
vlan internal allocation policy ascending
!
interface FastEthernet0/1
switchport mode access
dot1x port-control auto
spanning-tree portfast
!
interface FastEthernet0/2
!
interface FastEthernet0/3
!
interface FastEthernet0/4
!
interface FastEthernet0/5
!
interface FastEthernet0/6
!
interface FastEthernet0/7
!
interface FastEthernet0/8
!
interface FastEthernet0/24
!
interface GigabitEthernet0/1
```



```
!  
interface GigabitEthernet0/2  
!  
interface Vlan1  
ip address 192.168.3.240 255.255.255.0  
no ip route-cache  
!  
ip http server  
radius-server host 192.168.3.59 auth-port 1645 acct-port 1646 key  
alumno  
radius-server source-ports 1645-1646  
!  
control-plane  
!  
!  
line con 0  
line vty 5 15  
!  
!  
end
```

Odyssey Password:

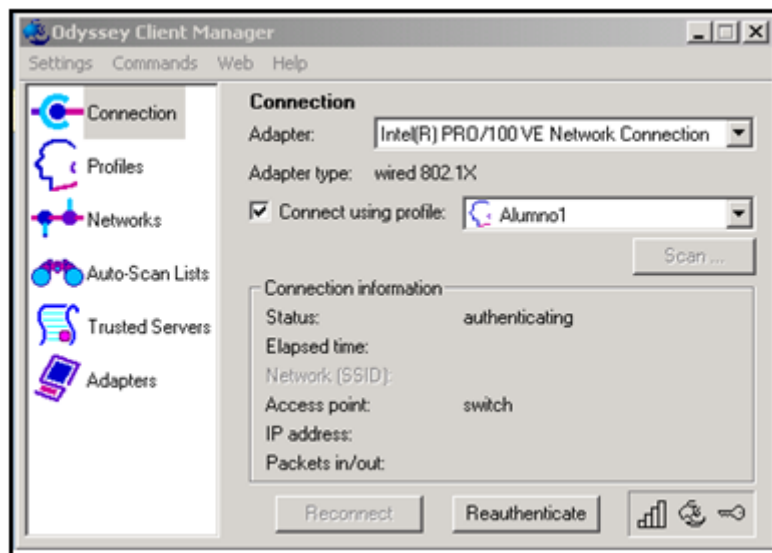


Figura 31. Pantalla de conexión.

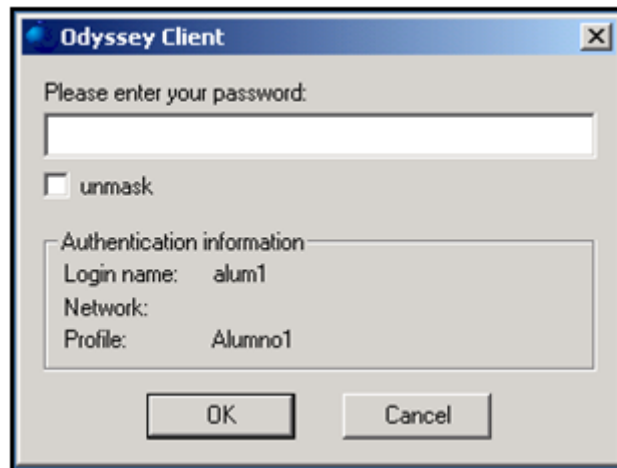


Figura 32. Autenticación.

Odyssey Autenticado:

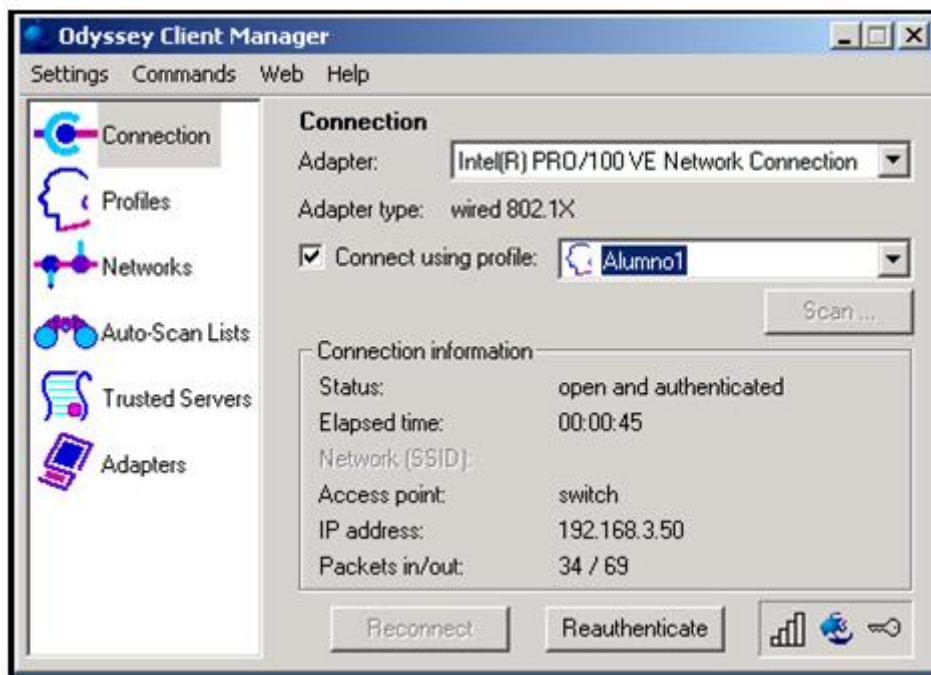


Figura 33. Conectado.

Realizamos un *ping* antes y después desde el suplicante a otro PC del laboratorio:

Antes:

```
C:\Documents and Settings\Administrador>ping 192.168.3.240
```

```
Haciendo ping a 192.168.3.240 con 32 bytes de datos:  
Tiempo de espera agotado para esta solicitud.
```




Tiempo de espera agotado para esta solicitud.
Tiempo de espera agotado para esta solicitud.

Estadísticas de ping para 192.168.3.240:
Paquetes: enviados = 4, recibidos = 0, perdidos = 4 (100% perdidos),
Tiempos aproximados de recorrido redondo en milisegundos:
mínimo = 0ms, máximo = 0ms, promedio = 0ms

Después:

```
C:\Documents and Settings\Administrador>ping 192.168.3.240
```

```
Haciendo ping a 192.168.3.240 con 32 bytes de datos:  
Tiempo de espera agotado para esta solicitud.  
Respuesta desde 192.168.3.240: bytes=32 tiempo<10ms TTL=255  
Respuesta desde 192.168.3.240: bytes=32 tiempo<10ms TTL=255  
Respuesta desde 192.168.3.240: bytes=32 tiempo<10ms TTL=255
```

Estadísticas de ping para 192.168.3.240:
Paquetes: enviados = 4, recibidos = 3, perdidos = 1 (25% perdidos),
Tiempos aproximados de recorrido redondo en milisegundos:
mínimo = 0ms, máximo = 0ms, promedio = 0ms

Ahora hacemos uso de la herramienta *Etherreal* para ver la secuencia que se intercambian entre el autenticador-servidor *RADIUS* y suplicante-Autenticador:

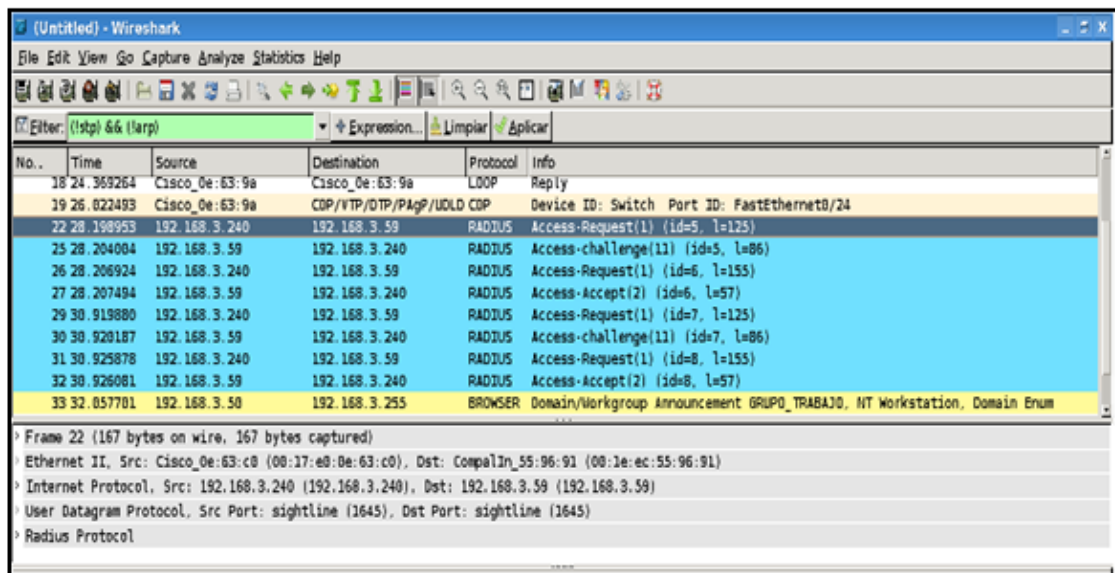


Figura 33. Secuencia que se intercambian entre el Autenticador-servidor *RADIUS* y suplicante-Autenticador.



CAPITULO 6. CERTIFICADOS DIGITALES PARA SERVIDORES WEB

6.1. Introducción

Debido a las carencias de seguridad del protocolo *HTTP* se implementa una capa adicional para el transporte de información que resuelve los aspectos de seguridad carentes en los servidores web. Esta capa viene definida por el estándar *TLS* o *Transport Layer Protocol* que gracias a las librerías *OpenSSL* permite cifrar la información a través de las conexiones seguras *SSL Secure Sockets Layer*.

SSL es un estándar definido por *Netscape Communication Corporation* y que actualmente implementan todos los servidores seguros. En el estándar *SSL* no solamente proporciona el cifrado y la confidencialidad de los datos, sino que también proporcionan autenticación de cliente y servidor y garantiza la integridad de los datos a través del protocolo *TCP/IP*.

Estas son las características deseables para que podamos considerar una conexión segura. Como la mayoría de servidores (entre ellos Apache) ya implementan estos estándares a través de las herramientas de *OpenSSL* sólo necesitamos de un certificado válido para poder establecer conexiones de tipo *HTTPS*.

6.1.1. Certificados digitales

Para solucionar el problema de autenticación en las transacciones por Internet, es necesario un sistema identificativo único de cada entidad o persona. Ya existen los sistemas criptográficos de clave asimétrica que se basan en el hecho de compartir una clave pública entre varios usuarios y otra privada, sólo conocida por el propietario. En general, esta clave se comparte mediante un directorio electrónico (normalmente en formato *LDAP*) o una página Web.

Sin embargo, este modo de compartir presenta un inconveniente importante ya que nada nos garantiza que la clave pertenezca al usuario con el que está asociada.

La solución a este problema condujo a la aparición de los **Certificados Digitales** o **Certificados electrónicos**, documentos electrónicos basados en la criptografía de clave pública y en el sistema de firmas digitales. Un certificado permite asociar una clave pública con una entidad (una persona, un equipo, etc....) para garantizar su validez. El certificado es como la tarjeta de identificación de la clave, emitida por una entidad llamada *Entidad de certificación* (que frecuentemente se abrevia **CA**, por sus siglas en inglés).

Las principales Autoridades Certificadoras actuales son:



- ❖ Verisign (www.verisign.com).
- ❖ Filial de RSA Data Security Inc.
- ❖ Thawte (www.thawte.com).
- ❖ La Fabrica Nacional de Moneda y Timbre (www.fnmt.es).

La entidad de certificación es responsable de emitir los certificados, de asignarles una fecha de validez y de revocarlos antes de esta fecha en el caso de que la clave (o su dueño) estén en una situación de riesgo.

El sistema es análogo a otros de uso común, como el D.N.I., en el que una autoridad de confianza (estado o policía) atestigua que la persona portadora de dicho documento es quién dice ser.

La estructura de los certificados está dividida en archivos pequeños, en dos partes:

- ❖ La parte que contiene la información.
- ❖ La parte que contiene la firma de la entidad de certificación.

La estructura de los certificados está estandarizada por la norma **X.509** (más precisamente, X.509v3) de la UIT, que define la información que contiene el certificado:

- ❖ El número de serie del certificado.
- ❖ El algoritmo de cifrado utilizado para firmar el certificado.
- ❖ El nombre (DN, siglas en inglés de *Nombre distinguido*) de la entidad de certificación que lo emite.
- ❖ La fecha en que entra en vigencia el certificado.
- ❖ La fecha en que finaliza el período de validez del certificado.
- ❖ El objeto de utilización de la clave pública.
- ❖ La clave pública del dueño del certificado.
- ❖ La firma del emisor del certificado (*huella digital*).

La entidad de certificación firma toda esta información (información + clave pública del solicitante) y esto implica que una **función hash** crea una huella digital de esta información y luego este **hash** se cifra con la clave privada de la entidad de certificación. La clave pública se distribuye antes de tiempo para permitir a los usuarios verificar la firma de la *entidad de certificación* con su clave pública.

Cuando un usuario desea comunicarse con otra persona, sólo debe obtener el certificado del receptor. Este certificado contiene el nombre y la clave pública del receptor, y está firmado por la entidad de certificación.

De esta forma, es posible verificar la validez del mensaje aplicando primero la función hash a la información contenida en el certificado y, segundo, descifrando la firma de la entidad de certificación con la clave pública y comparando los dos resultados.



6.1.2. Servidor Apache

La misión de un servidor Web es básicamente la de traducir una *URL* (*Uniform Resource Locator*) a un nombre de archivo y enviar este archivo a través de Internet, o a traducir la *URL* por el nombre de un programa, correr ese programa y enviar la respuesta de vuelta.

Cuando se conecta a una *URL*, lo que hace es enviar un mensaje a la máquina que tiene esa dirección. Lógicamente todos esperamos que la máquina a la que enviamos ese mensaje esté activa y lista para recibir y actuar sobre el mensaje enviado. Una *URL* tiene tres partes:

```
scheme://<host>/<path>
```

Así, si *<scheme>* es *http*, significa que el navegador debe utilizar el protocolo *http* (*Hypertext Transfer Protocol*). Si *<host>* es www.upct.es y *<path>* es */*, significa normalmente ir a la página superior de la máquina. La parte *<host>* puede contener una dirección IP o un nombre, que el navegador convertirá a una dirección IP. Las peticiones llegan al puerto 80 (puerto por defecto de HTTP) de la máquina *<host>*.

El servidor Apache (<http://www.apache.org>) es el servicio que se encarga de resolver las peticiones de las páginas de Internet de los clientes utilizando el protocolo de Internet *http*, como muestra la siguiente *Figura 34*.

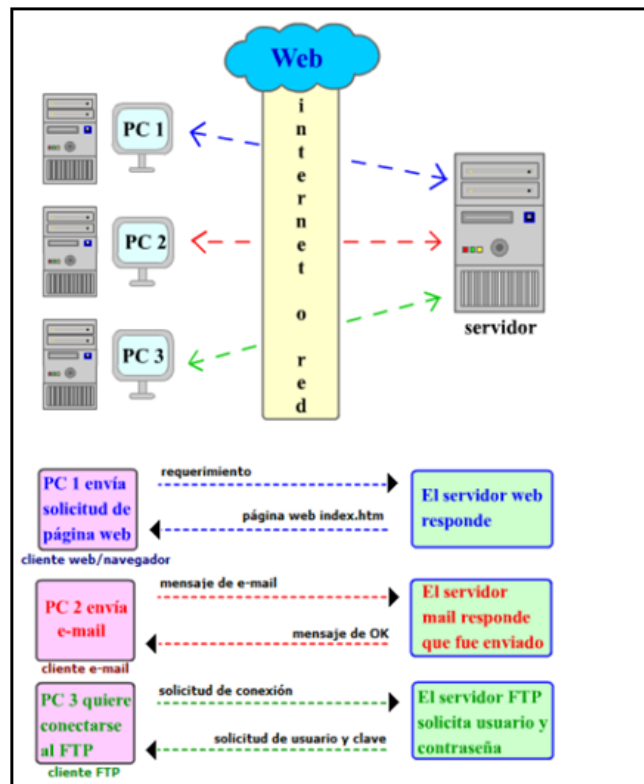


Figura 34. Servidor Apache.



Apache es un servidor Web mas utilizado en la actualidad en Internet, ocupando un lugar clave en la infraestructura de la red y cubre más de la mitad del mercado que se competidor directo *Microsoft*. Esto no solo es debido a que sea de libre distribución y por tanto gratuito, sino que su código es abierto, lo que significa que puede ser examinado por cualquiera, si hay errores miles de personas los detectan.

Este hecho lo hace más fiable que cualquier software comercial. En particular, Apache es extensible a través de una tecnología establecida para escribir nuevos módulos (Tomcat, mod_ssl, etc...), que permiten añadir nuevas características.

El modulo *SSL* que utilizaremos añade capacidades criptográficas al servidor Web Apache.

6.2. Instalación Servidor Web Apache + mod_SSL

Iniciamos el proceso de descarga del servidor web Apache <http://www.apache.org> (versión 1.3.41) y del paquete *mod_ssl* <http://www.modssl.org> (versión -2.8.31-1.3.41) estos dos números de serie, el primero (2.8.31) indica la versión del *mod_ssl*, el segundo (1.3.41) muestra la versión de Apache.

Descargamos el paquete *openssl* para Linux de <http://www.openssl.org> (versión 0.9.8h).El software *OpenSSL* es un proyecto de software desarrollado por lo miembros de la comunidad Open Source. Es un robusto juego de herramientas que le ayudan a su sistema a implementar el *Secure Sockets Layer* (SSL), así como otros protocolos relacionados con la seguridad, tales como el *Transport Layer Security* (TLS). También incluye una librería de criptografía. Este paquete de software es importante para cualquiera que esté planeando usar cierto nivel de seguridad en su máquina Linux.

Una vez descargado el servidor Web Apache, *mod_ssl* y *openssl*, será descomprimir nuestro servidor apache y configurarlo indicando en qué directorio lo queremos instalar:

```
#wget http://archive.apache.org/dist/httpd/apache_1.3.29.tar.gz
#tar xzf apache_1.3.29.tar.gz
#cd apache_1.3.29
#./configure --prefix=/usr/local/apache
#cd ..
```

A continuación instalaremos *Openssl* con los siguientes comandos. Tras descomprimir el archivo lo configuramos indicando dónde lo queremos instalar (opción *prefix* del *script config*), los compilamos y lo instalamos, como cualquier programa Linux que viene en forma de código fuente:



```
#wget http://openssl.org/source/openssl-0.9.7d.tar.gz
#tar xzf openssl-0.9.7d.tar.gz
#cd openssl-0.9.7d
#./config --prefix=/usr/local/ssl
#make
#make install
```

Descomprimos el *mod_SSL* que hemos descargado de Internet, y lo configuramos indicándole donde tenemos el código fuente del Servidor Apache (que aún no lo hemos instalado):

```
#wgethttp://www.modssl.org/source/OBSOLETE/mod_ssl-2.8.16-
1.3.29.tar.gz
#tar xzf mod_ssl-2.8.16-1.3.29.tar.gz
#cd mod_ssl-2.8.16-1.3.29
#./configure --with-apache=../apache_1.3.29
#cd ..
```

En este punto es donde podremos añadir a nuestro servidor Apache todos los módulos que queramos (Perl, PHP, Tomcat, ssl, etc.), de la misma forma que se indica en sus respectivas guías de instalación. En este caso, el único módulo a añadir es *mod_ssl*.

Primero le indicamos dónde hemos descomprimido *OpenSSL* y después le indicamos los distintos módulos que queremos usar:

```
#cd ../apache_1.3.29
#SSL_BASE=../openssl-0.9.7d ./configure --prefix=/usr/local/apache
--enable-module=ssl -enable- shared=ssl
```

Ya tenemos Apache preparado para instalarlo, así que compilamos:

```
#make
```



```
+-----+
| Before you install the package you now should prepare the SSL
| certificate system by running the 'make certificate' command.
| For different situations the following variants are provided:
|
| % make certificate TYPE=dummy (dummy self-signed Snake Oil cert)
| % make certificate TYPE=test (test cert signed by Snake Oil CA)
| % make certificate TYPE=custom (custom cert signed by own CA)
| % make certificate TYPE=existing (existing cert)
| CRT=/path/to/your.crt [KEY=/path/to/your.key]
|
| Use TYPE=dummy when you're a vendor package maintainer,
| the TYPE=test when you're an admin but want to do tests only,
| the TYPE=custom when you're an admin willing to run a real server
| and TYPE=existing when you're an admin who upgrades a server.
| (The default is TYPE=test)
|
| Additionally add ALGO=RSA (default) or ALGO=DSA to select
| the signature algorithm used for the generated certificate.
|
| Use 'make certificate VIEW=1' to display the generated data.
|
| Thanks for using Apache & mod_ssl. Ralf S. Engelschall
|                                     rse@engelschall.com
|                                     www.engelschall.com
+-----+
```

Ahora es el momento de crear nuestro certificado y clave, proceso realizado por la facilidad *mkcert.sh*. Será necesario contestar a diversas preguntas sobre nuestra “empresa”.

A toda esta información se le conoce como *Distinguished Name*. Algunas preguntas son opcionales (no llevan respuesta por defecto entre corchetes []). El Distinguished Name es requisito para crear una solicitud de certificado a cualquier Autoridad Certificadora como *VeriSign*.

Para crear el certificado y clave debemos usar el comando:

```
#make certificate
```

```
SSL Certificate Generation Utility (mkcert.sh)Copyright (c) 1998-2000
Ralf S. Engelschall, All Rights Reserved.
Generating test certificate signed by Snake Oil CA [TEST]
WARNING: Do not use this for real-life/production systems
```

```
STEP 0: Decide the signature algorithm used for certificate
The generated X.509 CA certificate can contain either
RSA or DSA based ingredients. Select the one you want to use.
Signature Algorithm ((R)SA or (D)SA) [R]:
```

```
STEP 1: Generating RSA private key (1024 bit) [server.key]
```



```
695820 semi-random bytes loaded
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
```

```
STEP 2: Generating X.509 certificate signing request [server.csr]
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

- ```

1. Country Name (2 letter code) [XY]:ES
2. State or Province Name (full name) [Snake Desert]:Murcia
3. Locality Name (eg, city) [Snake Town]:Cartagena
4. Organization Name (eg, company) [Snake Oil, Ltd]:IT4-PC00
5. Organizational Unit Name (eg, section) [Webserver Team]:src
6. Common Name (eg, FQDN) [www.snakeoil.dom]:www.IT4-PC00.upct.es
7. Email Address (eg, name@FQDN) [www@snakeoil.dom]:src@IT4-
PC00.upct.es
8. Certificate Validity (days) [365]:
```
- 

```
STEP 3: Generating X.509 certificate signed by Snake Oil CA
[server.crt]
Certificate Version (1 or 3) [3]:
Signature ok
subject=/C=ES/ST=Murcia/L=Cartagena/O=IT4-PC00/OU=src/CN=www.IT4-
PC00.upct.es/emailAddress=src@IT4-PC00.upct.es
Getting CA Private Key
Verify: matching certificate & key modulus
Verify: matching certificate signature
../conf/ssl.crt/server.crt: /C=XY/ST=Snake Desert/L=Snake Town/O=Snake
Oil, Ltd/OU=Certificate Authority/CN=Snake Oil
CA/emailAddress=ca@snakeoil.dom
OK
```

---

```
STEP 4: Encrypting RSA private key with a pass phrase for security
[server.key]
The contents of the server.key file (the generated private key) has to
be kept secret. So we strongly recommend you to encrypt the server.key
file with a Triple-DES cipher and a Pass Phrase.
Encrypt the private key now? [Y/n]:
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

---





Fine, you're using an encrypted RSA private key.

---

RESULT: Server Certification Files

o conf/ssl.key/server.key

The PEM-encoded RSA private key file which you configure with the 'SSLCertificateKeyFile' directive (automatically done when you install via APACI). KEEP THIS FILE PRIVATE!

o conf/ssl.crt/server.crt

The PEM-encoded X.509 certificate file which you configure with the 'SSLCertificateFile' directive (automatically done when you install via APACI).

o conf/ssl.csr/server.csr

The PEM-encoded X.509 certificate signing request file which you can send to an official Certificate Authority (CA) in order to request a real server certificate (signed by this CA instead of our demonstration-only Snake Oil CA) which later can replace the conf/ssl.crt/server.crt file.

El directorio donde se guardan el archivo de clave privada es → conf/ssl.key/server.key

El directorio donde se guardan el archivo con el certificado es → conf/ssl.key/server.crt

El directorio donde se guardan la solicitud del certificado es → conf/ssl.key/server.csr

Según la información mostrada en la pantalla después de compilar explicamos el número de opciones que tenemos para crear un certificado y el significado de cada una de ellas:

- ❖ make certificate TYPE=dummy → Autofirmado.
- ❖ make certificate TYPE=test → Firmado por una Autoridad Certificadora.
- ❖ make certificate TYPE=custom → Firmado por una Autoridad Certificadora existente.
- ❖ make certificate TYPE=existing → Utilizamos un certificado existente.

Por último queda instalar los archivos en su lugar correspondiente:

```
#make install
```



```
+-----+
| You now have successfully built and installed the
| Apache 1.3 HTTP server. To verify that Apache actually
| works correctly you now should first check the
| (initially created or preserved) configuration files
|
| /usr/local/apache/conf/httpd.conf
|
| and then you should be able to immediately fire up
| Apache the first time by running:
|
| /usr/local/apache/bin/apachectl start
|
| Or when you want to run it with SSL enabled use:
|
| /usr/local/apache/bin/apachectl startssl
|
| Thanks for using Apache. The Apache Group
| http://www.apache.org/
+-----+
```

Como podemos observar la instalación ha sido satisfactoria y podemos ver los archivos de configuración donde se encuentran y las opciones que tenemos para instalar un servidor:

```
#/usr/local/apache/bin/apachectl start → Modo normal
#/usr/local/apache/bin/apachectl startssl → Modo seguro
```

Arrancamos la versión segura, la cuál, nos pedirá el password que hallamos dado en nuestra clave SSL en nuestro caso (*mysuperpassword*) y a partir de este momento podemos acceder a nuestro servidor seguro desde nuestro navegador.

```
#/usr/local/apache/bin/apachectl startssl
```

```
[Tue Sep 9 15:30:30 2008] [alert] httpd: Could not determine the
server's fully qualified domain name, using 127.0.0.1 for ServerName
```

```
Apache/1.3.41 mod_ssl/2.8.31 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide us with the pass phrases.
```

```
Server localhost.upct.es:443 (RSA)
Enter pass phrase: mysuperpassword
```

```
Ok: Pass Phrase Dialog successful.
/usr/local/apache/bin/apachectl startssl: httpd started
```

Para comprobar que funciona correctamente dirigimos nuestro navegador a la dirección IP de la máquina donde estamos con el *[scheme]* a https y recibiremos una ventana informándonos del acceso a “zona segura” y de que el certificado no está aprobado por ningún organismo oficial, por lo que podría no ser válido como se puede ver en la *Figura X*:

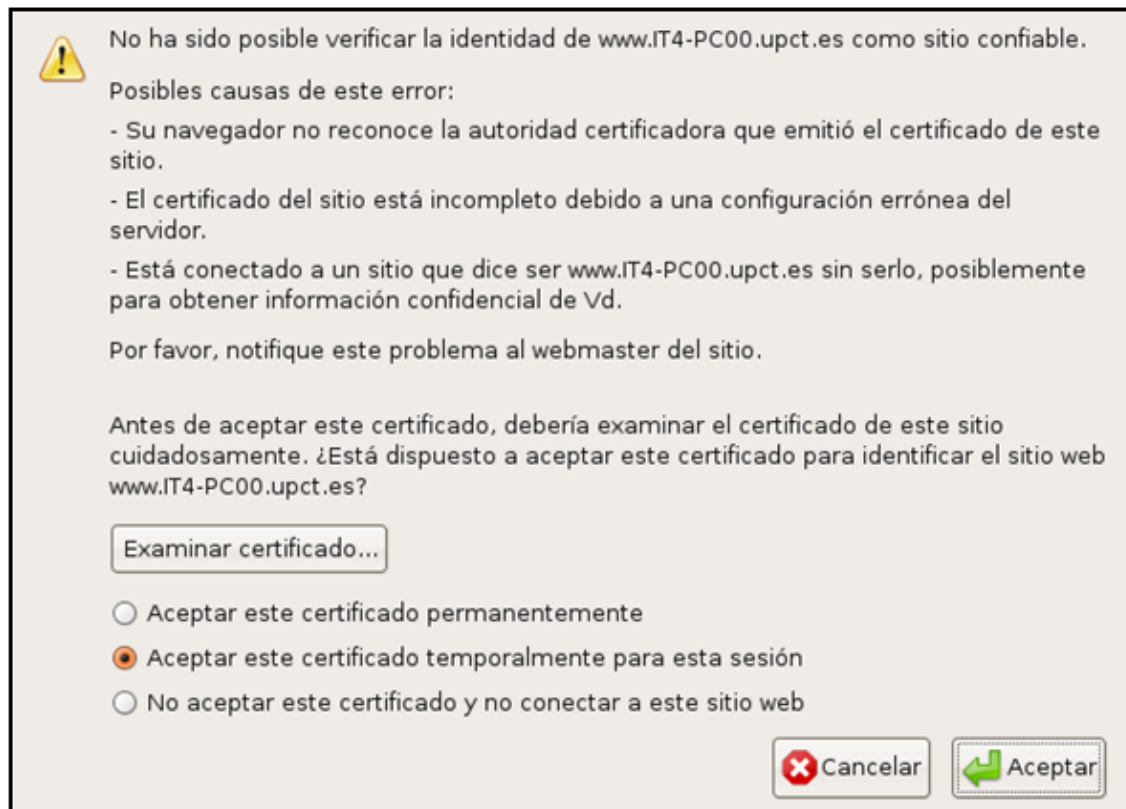


Figura 35. Acceso a zona segura.

Aceptamos el certificado temporal para esta sesión.

Aquí vemos los detalles del certificado con cada una de sus características:



Figura 36. Certificado.

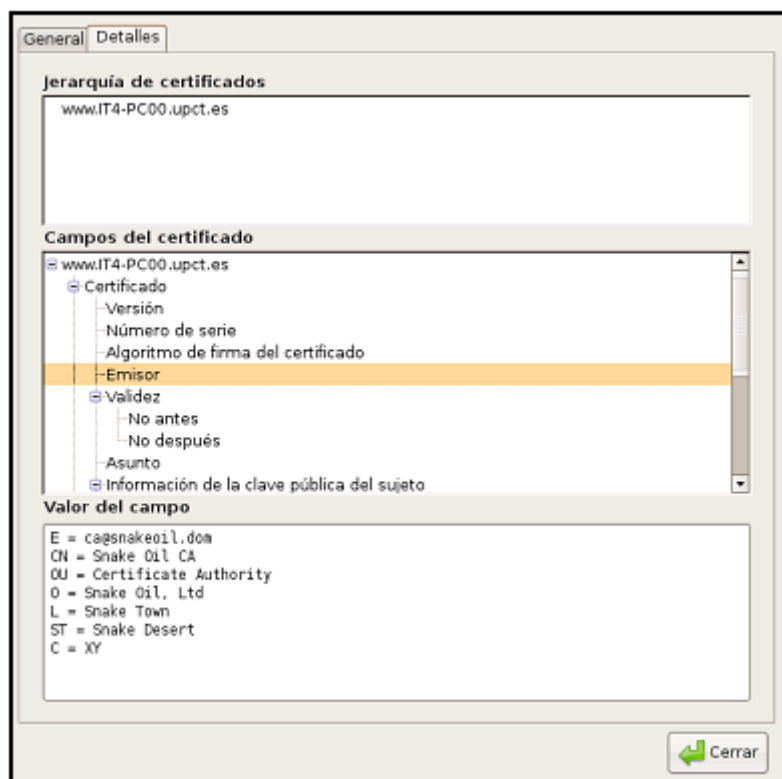


Figura 37. Detalles del certificado.



Tras aceptar el certificado (**solo para esta sesión**) nos sale la página de presentación de Apache con la información de SSL.



Figura 38. Presentación de Apache con la información de SSL.

### 6.3. Instalación del Certificado VeriSign

Para solicitar un certificado firmado por una autoridad certificadora nos dirigimos a la URL <https://www.verisign.com/products/srv/trial/intro.html> de VeriSign.

VeriSign es la empresa fundada en 1995 por RSA Data Security System y otros socios, líder en emisión de certificados SSL (*Secure Sockets Layer*) que permiten las comunicaciones y el comercio electrónico seguro en sitios Web, intranets y extranets.

Desde la URL solicitaremos un certificado temporal y lo instalaremos en el servidor Apache.

- Cree una solicitud de Certificado de prueba en el administrador de claves y regístrese para obtener un Certificado de prueba con Verisign. Asegúrese de utilizar una contraseña de 7 caracteres o menos.
- Recibirá un certificado vía correo electrónico con el siguiente aspecto:





```
/usr/local/apache/conf/ssl.crt/verisign.crt
```

**Nota:** Aquí hemos guardado el nuevo certificado temporal en *verisign.crt*.

Ahora sólo nos queda guardar este archivo en el archivo de configuración indicándole la nueva ruta donde se encuentra el certificado firmado por la autoridad y la clave privada en:

```
SSLCertificateFile en /usr/local/apache/conf/httpd.conf
```

Por:

```
SSLCertificateFile en /usr/local/apache/conf/verisign.crt
```

Asegúrese que no hay espacios al principio o al final de cada línea del certificado. Muchos sistemas de correo electrónico cambian el formato del documento. Este paso es esencial para evitar errores de formato. Reiniciamos el servidor Apache:

```
#!/usr/local/apache/bin/apachectl restart
```

Aquí vemos en la *Figura 38*, los detalles del certificado con cada una de sus características:



Figura 38. Certificado.

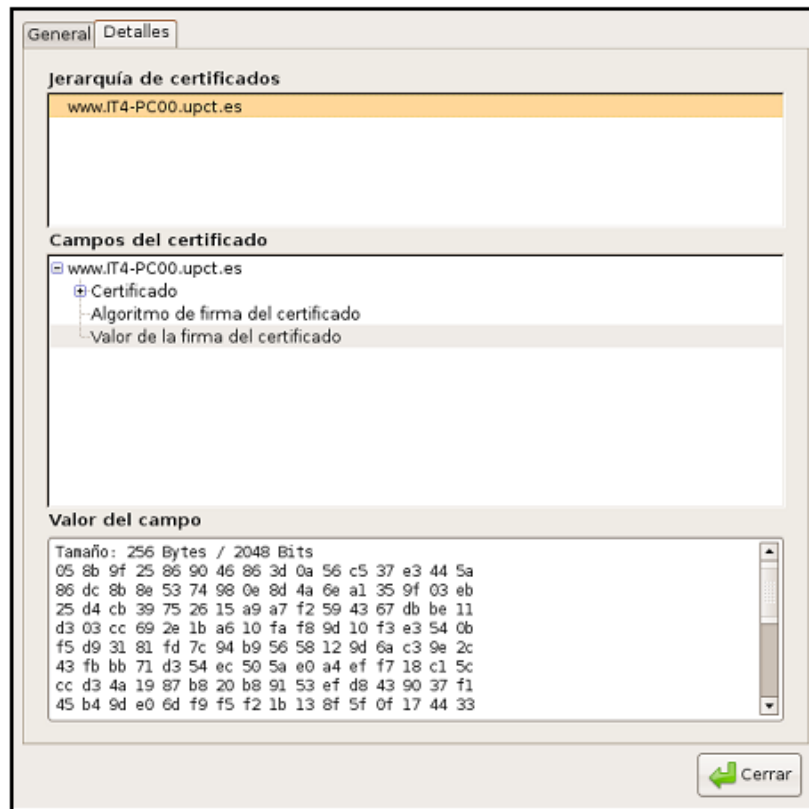


Figura 39. Detalles del certificado.

Tras aceptar el certificado nos sale la página de presentación de Apache con la información de SSL.



Figura 40. Presentación de Apache con la información SSL.





## CAPITULO 7. INSTALACIÓN Y MANEJO DE PGP

### 7.1. Introducción

*PGP (Pretty Good Privacy)* es un criptosistema (sistema de cifrado) inventado por *Philip Zimmermann*, un analista de sistemas en 1991. Es extremadamente rápido y fiable. Desde su aparición se ha convertido en una de las herramientas más utilizadas a nivel mundial para conseguir privacidad y autenticación tanto en los mensajes de correo como en los archivos almacenados en el disco duro de su ordenador. A ello a contribuido indudablemente su distribución como herramienta gratuita, así como su puesta al día en sucesivas versiones aparecidas mejorando los algoritmos criptográficos utilizados. *PGP* se puede encontrar como *plug-in* para la mayoría de agentes de usuario de correo electrónico, incluyendo *Exchange* y *Outlook de Microsoft*, *Eudora* o *Pine*. *PGP* es un sistema de criptografía *híbrido* que usa una combinación de funciones tomadas de la criptografía de clave pública y de la criptografía simétrica.

*PGP* nos permite dos cosas:

- ❖ Cifrar mensajes y archivos para que no resulten legibles sin nuestra autorización (**Confidencialidad**).
- ❖ Firmarlos digitalmente para asegurarnos que no son modificados sin nuestro consentimiento (**Integridad y autenticación**).

Cuando un usuario cifra un texto con *PGP*, los datos primero se comprimen. Esta compresión de datos permite reducir el tiempo de transmisión a través del canal de comunicación, ahorra espacio en el disco duro y, lo más importante, aumenta la seguridad criptográfica.

La mayoría de los criptoanalistas sacan provecho de los modelos encontrados en formato de sólo texto para descubrir el cifrado. La compresión reduce estos modelos de sólo texto y mejora considerablemente su resistencia a los criptoanalistas.

El cifrado se realiza, principalmente, en dos fases:

1. *PGP* crea una clave secreta *IDEA* en forma aleatoria y cifra los datos con esta clave.
2. *PGP* cifra la clave secreta *IDEA* y la envía usando la clave pública *RSA* del receptor.

El descifrado también se reduce en dos fases:

1. *PGP* descifra la clave secreta *IDEA* usando la clave privada *RSA*.
2. *PGP* descifra los datos con la clave secreta *IDEA* obtenida previamente.



El método de cifrado combina la fácil utilización del cifrado de clave pública con la velocidad del cifrado convencional que es aproximadamente 1000 veces más rápido que los algoritmos de cifrado de clave pública. El cifrado de clave pública resuelve el problema de la distribución de la clave. Combinados, estos dos métodos mejoran el rendimiento y administración de las claves sin poner en peligro la seguridad.

## 7.2. Funciones y Servicios de PGP

PGP ofrece las siguientes funciones *Figura 41*.

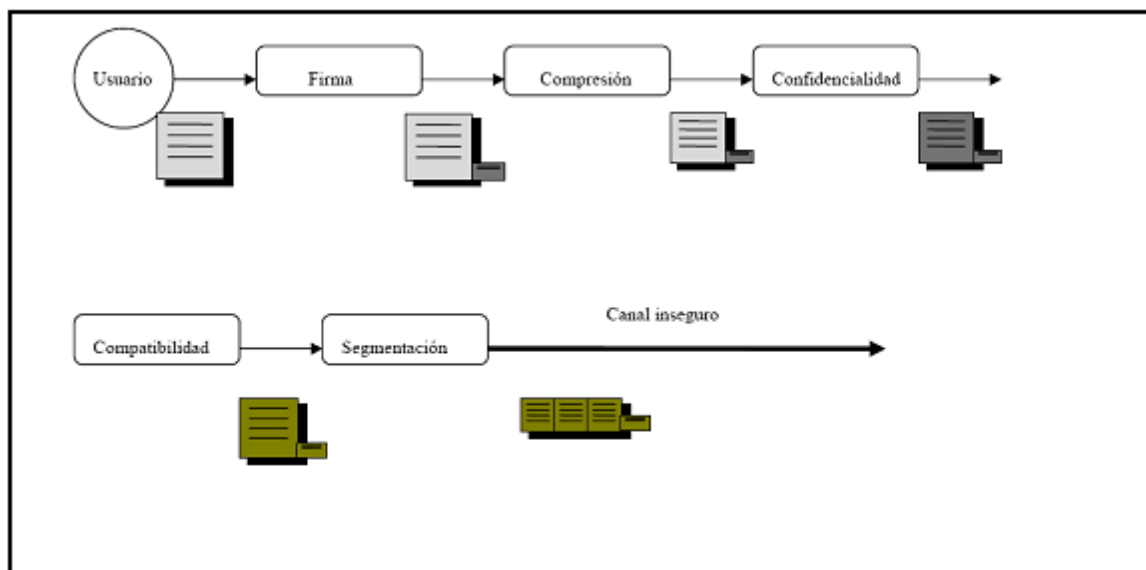


Figura 41. Funciones.

- ❖ **Firmas digitales y verificación de la integridad de los mensajes:** función que se basa en el uso simultáneo de la función *hash* (MD5) y del sistema *RSA*. La función *MD5* condensa el mensaje y produce un resultado de 128bits que después se cifra, gracias al algoritmo *RSA*, por la clave privada del emisor.
- ❖ **Cifrado de archivos locales:** función que utiliza el algoritmo *IDEA*.
- ❖ **Generación de claves públicas o privadas:** cada usuario cifra su mensaje mediante las claves privadas *IDEA*. La transferencia de las claves electrónicas *IDEA* utiliza el sistema *RS*. Por lo tanto, *PGP* ofrece dispositivos de generación de claves adaptados al sistema. El tamaño de las claves *RSA* se propone de acuerdo con varios niveles de seguridad: 512, 768, 1024 ó 1280 bits.
- ❖ **Administración de claves:** función responsable de la distribución de la clave pública del usuario a los remitentes que desean enviarle mensajes cifrados.



- ❖ **Certificación de claves:** esta función permite agregar un sello digital que garantice la autenticidad de las claves públicas. Es una característica original de *PGP*, que basa su confianza en una noción de proximidad social en vez de en una entidad certificación central.
- ❖ **Revocación, desactivación y registro de claves:** función que permite producir certificados de revocación.

*PGP* presenta los siguientes servicios:

| Servicios        | Algoritmos utilizados |
|------------------|-----------------------|
| Confidencialidad | IDEA y RSA            |
| Firma            | RSA y MD5 o DSS       |
| Compresión       | ZIP                   |
| Compatibilidad   | Radix 64              |
| Segmentación     |                       |

Tabla 10. Servicios.

## 7.3. Descripción

### 7.3.1. Notación

$K_s$  = clave secreta o de sesión usada en el esquema de cifrado simétrico.

$K_{Ra}$  = clave privada del usuario A.

$K_{Ua}$  = clave pública del usuario A.

EP = cifrado de clave pública.

DP = descifrado de clave pública.

EC = cifrado simétrico.

DC = descifrado simétrico.

H = función *hash*.

|| = concatenación.

Z = compresión utilizando el algoritmo ZIP.

R64 = conversión al formato ASCII radix 64

### 7.3.2. Firma Digital o Autenticación

Como puede verse en la *Figura 42*, el servicio de firma se proporciona de la siguiente manera:

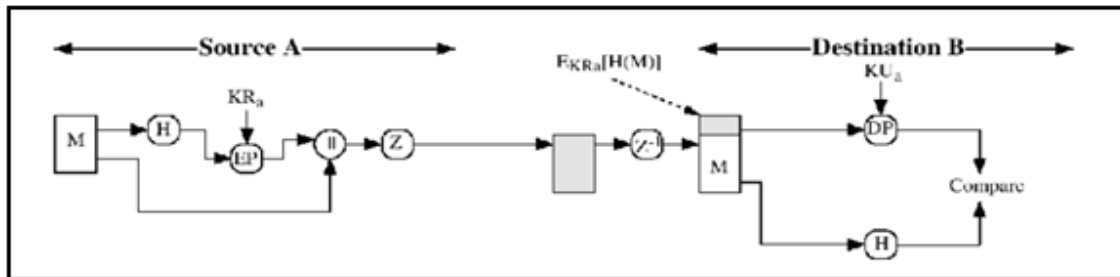


Figura 42. Autenticación.

La secuencia seguida en este caso es la siguiente:

1. El emisor crea un mensaje.
2. Se usa *SHA-1* para generar un código *hash* del mensaje de 160 bits.
3. El código *hash* se cifra con *RSA* usando la clave privada del emisor y el resultado se añade antepuesto al mensaje.
4. La firma junto con el mensaje se comprime y se transmite.
5. El receptor descomprime el mensaje y usa *RSA* con la clave pública del emisor para descifrar y recuperar el código *hash*.
6. El receptor genera un nuevo código hash para el mensaje y lo compara con el código **hash** descifrado. Si los dos coinciden, el mensaje se considera auténtico y se acepta.

### 7.3.3. Confidencialidad

El servicio de confidencialidad se utiliza tanto para cifrar mensajes que se van a transmitir o almacenarse localmente en ficheros. En ambos casos se puede utilizar el algoritmo de cifrado simétrico *CAST-128* y, como alternativa, *IDEA* o *3DES*, utilizando siempre el modo CFB de 64 bits.

Como puede verse en la *Figura 43*, la confidencialidad se proporciona de la siguiente manera:

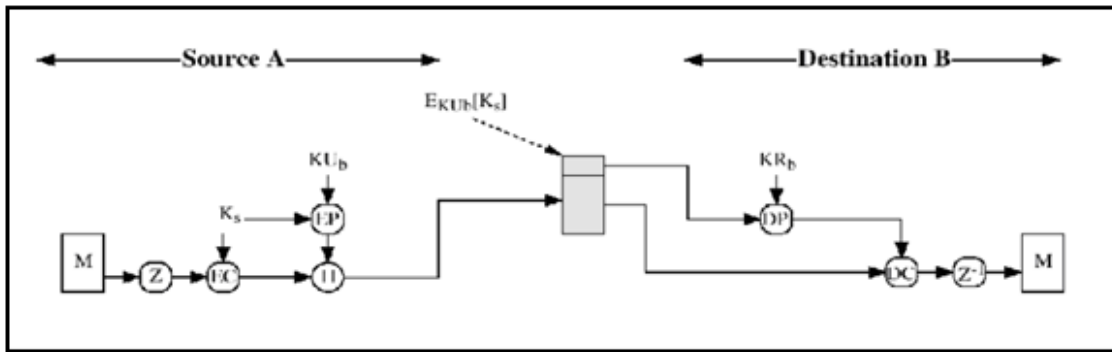


Figura 43. Confidencialidad.

La secuencia seguida en este caso es la siguiente:

1. El emisor genera un mensaje y *PGP* lo comprime.
2. El mensaje se cifra, usando *CAST-128* (o *IDEA* o *3DES*) con la clave secreta o de sesión.
3. La clave de sesión se cifra con *RSA*, usando la clave pública del receptor, y se añade antepuesta el mensaje.
4. El receptor usa *RSA* con una clave privada para descifrar y recuperar la clave de sesión.
5. La clave de sesión se usa para descifrar el mensaje comprimido.
6. El mensaje ya en texto claro se descomprime.

### 7.3.4. Autenticación y Confidencialidad

Ambos servicios se pueden utilizar con un mismo mensaje. En este caso, primero se genera una firma para el mensaje en texto claro y se adjunta antepuesta a dicho mensaje. El resultado (mensaje + firma) se comprime y se cifra usando la clave de sesión. Por último, la clave de sesión se cifra usando *RSA*, con la clave pública del receptor.

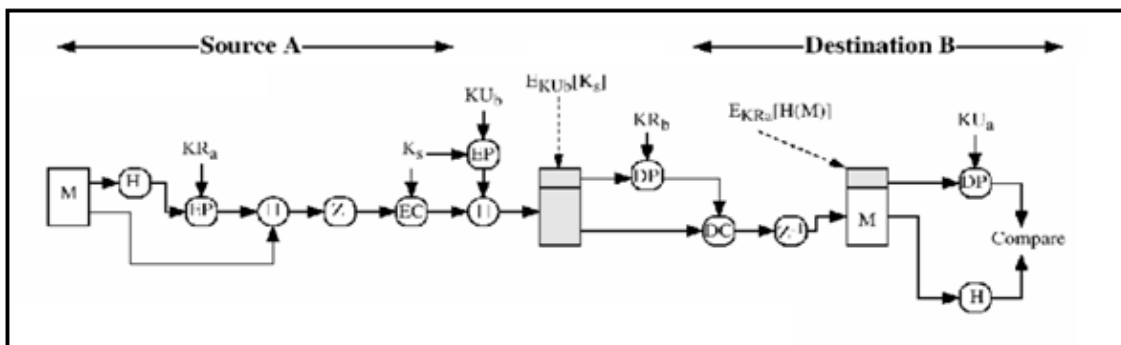


Figura 44. Autenticación y Confidencialidad.



### 7.3.5. Compresión

De forma predeterminada, *PGP* comprime el mensaje después de aplicar la firma, pero antes del cifrado.

La firma se genera antes de la compresión debido a dos razones fundamentales:

- ❖ Es preferible firmar un mensaje descomprimido para poder almacenar solamente el mensaje descomprimido junto con la firma para su verificación posterior. Si se firma un documento comprimido, sería necesario almacenar una versión comprimida del mensaje para su posterior verificación o volver a comprimir el mensaje cuando se requiera verificación.
- ❖ El algoritmo de compresión no es determinista, distintas implementaciones del algoritmo permiten diferentes compromisos entre la velocidad de ejecución y el ratio de compresión y, como resultado, producen distintas secuencias comprimidas. Sin embargo. Los distintos algoritmos de compresión pueden operar entre sí, ya que cualquier versión del algoritmo puede descomprimir correctamente la salida de cualquier otra versión. Aplicar la función *hash* y la firma después de la compresión obligaría a utilizar siempre la misma versión del algoritmo de compresión.

El algoritmo de compresión que se utiliza es *ZIP*.

El algoritmo que se utiliza para la encriptación no es siempre con *RSA*, debido a que el algoritmo es muy lento y, además, las claves se deben utilizar con el mínimo texto posible, o sea, cambiarlas a menudo. La generación de pares de claves pública/privada es un proceso muy lento (del orden de minutos), por lo tanto no es práctico generarlas a cada sesión. Aún así las claves públicas utilizadas en la firma y en la encriptación de la clave de sesión se debe cambiar cada cierto periodo de tiempo.

### 7.3.6. Confidencialidad con correo electrónico

Muchos sistemas de correo electrónico sólo permiten caracteres *ASCII* de 7 bits. Esto es un problema para transmitir ciertos símbolos y letras utilizados normalmente en los documentos, especialmente en los idiomas latinos, nórdicos o eslavos donde se acentúan las palabras. Para solucionar esto se utiliza un sistema para mapear los símbolos de 8 bits en palabras de 7 bits.

El sistema utilizado por *PGP* se denomina *RADIX-64*.

*RADIX 64* realiza la transformación indicada en la *Figura 45*.

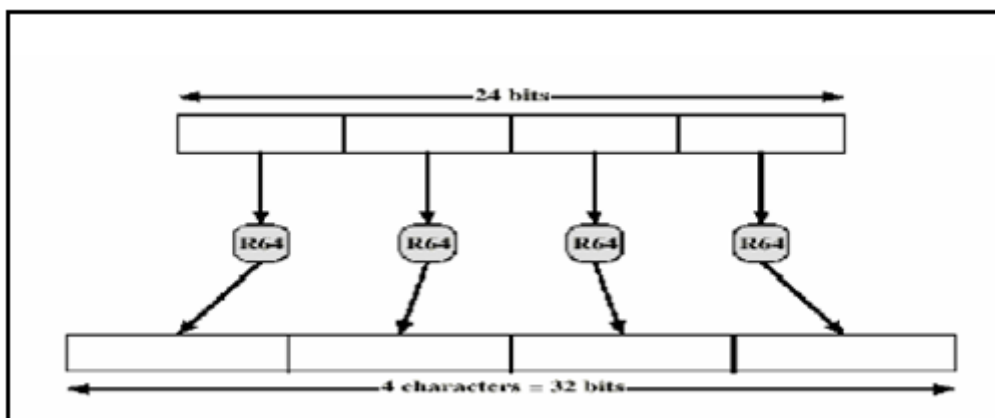


Figura 45. Radix 64.

Esta transformación produce una expansión del fichero del 33% pero, normalmente, se compensa con compresiones del algoritmo ZIP de más del 50%. Esta transformación debe ser la última, ya que las anteriores siempre trabajan con palabras de 8 bits.

### 7.3.7. Segmentación

La mayoría de sistemas *E-MAIL* no permiten ficheros mayores que una longitud máxima. Para evitar problemas en el envío de ficheros de gran tamaño, *PGP* ofrece este servicio de segmentación en la emisión y ensamblar en la recepción. Este proceso siempre es el último de la cadena.

## 7.4. Gestión de Claves

La gestión de claves es un proceso muy importante para todos los sistemas de seguridad, por muy segura que sea la criptología aplicada no sirve de nada si al distribuir las claves el criptoanalista puede interceptarlas, sustituirlas, etc.

En *PGP*, el mayor problema de distribución es para las claves simétricas, la solución está en añadirlas al mensaje cifradas con RSA, por lo tanto, su confidencialidad está asegurada por la robustez del algoritmo asimétrico.

El sistema de distribución de claves públicas es libre, o sea, cada usuario se encarga de la distribución de su clave: por teléfono, personalmente, por Internet, etc. Pero, excepto en el caso de entregarla personalmente, todos estos métodos libres son vulnerables de una suplantación de personalidad.

Para evitar la suplantación de personalidad se utiliza los certificados de clave pública.



Los certificados son mensajes con la siguiente información:

- ❖ Identidad del usuario.
- ❖ Clave pública del usuario.
- ❖ Fecha de creación y caducidad.
- ❖ Otras informaciones.
- ❖ Firma de una tercera persona de confianza.

La seguridad de que la clave pública pertenece al usuario depende de la confianza de la tercera persona, así la firma asegura la relación entre la clave y nombre del usuario. Si no se confía en la tercera persona, el certificado no sirve. Para solucionar el problema de terceras personas fiables se han creado unas empresas/instituciones denominadas autoridades de certificación (CA). Las CA firman certificados de personas que se han identificado, por lo tanto, cumplen la función de tercera persona de confianza, son como el estado respecto a los documentos de identidad (DNI).

PGP no está preparado para trabajar con CA, solamente para la certificación de claves públicas entre usuarios. Cada usuario que ha entregado su clave pública puede certificar también las claves de otros en los que confía.

Las claves públicas y privadas se almacenan en dos bases de datos de la máquina cliente del usuario, éstas son:

- ❖ **Anillos de clave privada.** Almacena encriptados mediante el algoritmo *IDEA* los pares de claves privada/pública propios. El *password* de usuario es la clave secreta para descifrar este anillo. Para evitar ataques a este *password* nunca se almacena en la máquina y se utiliza *passphrase* (*password* de una frase que se comprimen para 128 bits mediante una función *Hash*).
- ❖ **Anillos de clave pública.** Almacena las claves públicas de otros usuarios. Los datos de cada registro son:
  - Identificador de usuario.
  - Clave pública.
  - Nivel de confianza propio otorgado por el propietario del anillo a esta clave del usuario (depende de la fuente).
  - Certificaciones de la clave enviadas por otros usuarios.
  - Indicador de confianza de la clave. Es la confianza de la clave para firmar otros certificados.

El PGP permite utilizar varios pares de claves privadas/públicas por persona. Así se puede cambiar frecuentemente y mantener durante un periodo de tiempo las dos vigentes, así como, utilizar diferentes pares de claves para diferentes receptores (la empresa, anónimos de Internet, etc.). Para evitar confusiones con diferentes claves del mismo usuario, se envía con el mensaje un campo con los 64 primeros octetos de la clave pública utilizada, tanto en los servicios de confidencialidad como en los de firma.





También existe la posibilidad de que un usuario quiera revocar su clave pública actual porque sospecha que existe un riesgo o simplemente para evitar el uso de la misma clave durante un periodo largo de tiempo.

El convenio para revocar una clave pública es que el propietario emita un certificado de revocación de clave, firmado por él. Este certificado tiene la misma forma que un certificado de firma normal pero incluye un indicador que señala que su propósito es revocar la clave pública que contiene.

Es aconsejable generar un certificado de revocación, cuando se genera para claves pública y privada. De esta forma, si se nos olvida la frase clave, o perdemos la clave privada, siempre podemos revocar la clave pública.

## 7.5. Instalación

El programa viene comprimido con zip, así que lo primero será descomprimirlo.



Figura 46. Instalación.

Una vez descomprimido, vamos a la carpeta donde lo hemos hecho y ejecutamos Setup.exe para comenzar la instalación. La primera pantalla es de Bienvenida y nos aconseja que cerremos todos aquellos programas que se estén ejecutando para realizar la instalación. Pulsamos sobre **Next**.



Figura 47. Instalación.



Pantalla con la licencia. Pulsamos **yes**.

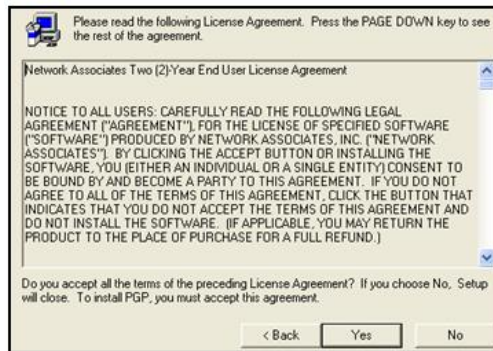


Figura 48. Instalación.

A continuación nos pregunta por el directorio donde se va a instalar *PGP*. Podemos cambiarlo (pulsando sobre *BROWSE*) o dejarlo por defecto donde esta. Pulsamos sobre **next**.



Figura 49. Instalación.

Ahora nos pide que seleccionemos del paquete, que componentes vamos a instalar. Una vez hecho pulsamos sobre **next**.



Figura 50. Instalación.

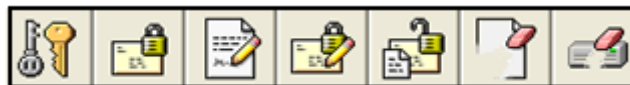


Esta Pantalla nos ofrece información sobre lo que le hemos indicado que debe de hacer para instalarse. Lo revisamos a ver si es correcto (en caso que no lo sea pulsar BACK para ir hacia atrás en las pantallas y corregir lo que queramos). Pulsamos, si todo es correcto, sobre **NEXT**.



**Figura 51. Instalación.**

Una vez finalizada la instalación, lanzamos la aplicación PGPTools. Esta aplicación incorpora los siguientes módulos, a los que se puede acceder directamente desde su botonera:



**Figura 52. PGPTools.**

Los módulos son los siguientes:

- ❖ *PGPKeys* (gestor de claves).
- ❖ Cifrado de archivos.
- ❖ Firma digital de archivos.
- ❖ Firma y cifrado simultáneo de archivos.
- ❖ Descifrado de archivos y verificación de firmas.
- ❖ Borrado de archivos.
- ❖ Borrado de espacio libre en el disco.

### ***7.5.1. Gestión de claves Pública / Privada***

Ahora viene el proceso de creación de nuestras claves, privada y pública. Pulsamos sobre **siguiente**.



Figura 53. Instalación.

Tenemos que asociar las claves con nuestro nombre y dirección de correo electrónico. Rellenamos ambos campos y pulsamos **siguiente**.



Figura 54. Instalación.

Aquí marcamos el tipo:



Figura 55. Instalación.



Figura 56. Instalación.



Figura 57. Instalación.

Una medida de protección más, es que tenemos que proteger con una clave, que solo conozcamos nosotros, nuestra clave privada. En este caso no está hablando de una palabra clave, sino que escribamos una frase. Cuanto más larga sea (mínimo 8 letras) y menos obvia, más segura será. Para ello debemos de escribir en *Passphrase* la frase o palabra (de más de 8 letras) y volver a escribirla en *Confirmation* para comprobar que no nos hemos equivocado. Podemos desmarcar la opción de *Hide Typing* para ver lo que estamos escribiendo. Una vez hecho esto pulsar sobre **siguiente**.



Figura 58. Instalación.



Figura 59. Instalación.

Una vez finalizada la instalación y reiniciado el ordenador, podemos comprobar que en la parte inferior derecha de la pantalla comprobamos que PGP está funcionando correctamente:



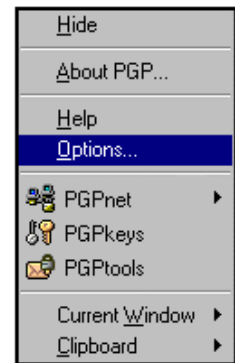
Figura 60. Instalación.



## 7.6. Configuración

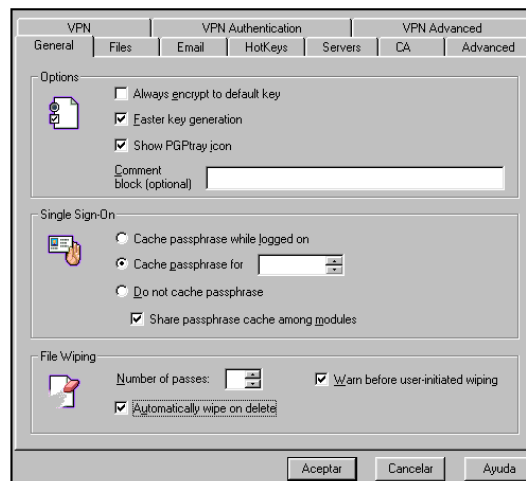
La configuración de los distintos parámetros de PGP se nos muestra en el menú principal de opciones. Las opciones disponibles que se muestra en la *Figura 61*.

|                  |                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------|
| <b>Hide</b>      | Desaparece de la pantalla en icono de PGP.                                                    |
| <b>About PGP</b> | Nos aparece una pantalla con información sobre PGP.                                           |
| <b>Help</b>      | Nos Abrió una ventana con la ayuda disponible para PGP.                                       |
| <b>Options</b>   | Está pestaña es la que más nos interesa y explicaremos las distintas opciones a continuación. |



**Figura 61. Configuración.**

Dentro de **options** se encuentra las siguientes pestañas más importantes *Figura 62*.



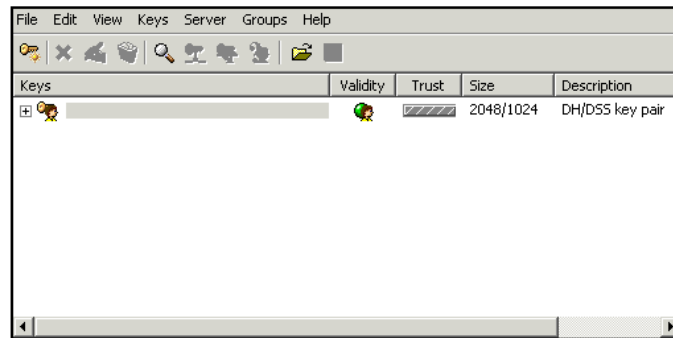
**Figura 62. Configuración.**

La descripción de cada una de las pestañas es:

- ❖ **General:** Nos solicita las opciones de cifrado y firma.
- ❖ **File:** Donde se encuentran las claves y ficheros que vamos a utilizar.
- ❖ **Email:** Opciones para uso de PGP con tu cliente de correo.
- ❖ **Hotkeys:** Podemos utilizar el teclado para realizar distintas tareas de PGP.
- ❖ **Servers:** Son los servidores que podemos utilizar para sincronizar las claves públicas (nuestra y la de los demás usuarios).
- ❖ **Ca:** Certificados de Autoridad, sirve para conocer el origen de una clave y si es válida.
- ❖ **Advanced:** Opciones avanzadas para elegir algoritmos criptográficos.

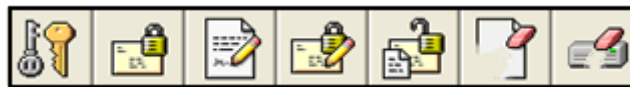


Una de las opciones importantes del menú principal es **PGPKeys**, nos aparecen las claves que tenemos en nuestro ordenador o para poder importar alguna clave. Para ello pulsamos sobre *Keys*, después sobre *Import* y nos pedirá que especifiquemos donde está el fichero con la clave que nos han enviado, como se muestra en la *Figura 63*.










**Figura 63. PGPKeys.**

Otra opción del menú principal es la de **PGPtools**. Pulsando sobre ella nos aparecerá una ventana para realizar diferentes tareas *Figura 64*.



**Figura 64. PGPtools.**

|                                                                                     |                                                                                                         |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
|  | Abre la ventana de PGPKeys                                                                              |
|  | Cifra un fichero                                                                                        |
|  | Firma un fichero                                                                                        |
|  | Cifra y firma un fichero                                                                                |
|  | Descifra o verifica un fichero                                                                          |
|  | Borra un fichero de forma segura                                                                        |
|  | Se borra todo el espacio libre de vuestro disco duro, para eliminar información que se encuentre en él. |

**Tabla 11. PGPtools.**



## 7.7. Funcionamiento

El funcionamiento de PGP es muy amplio, nosotros vamos a realizar dos opciones de las que nos ofrece el programa, en primer lugar explicaremos en términos generales el funcionamiento para poder cifrar ficheros, y en segundo lugar el funcionamiento con el programa de correo Outlook Express

### 7.7.1. Funcionamiento para cifrar ficheros

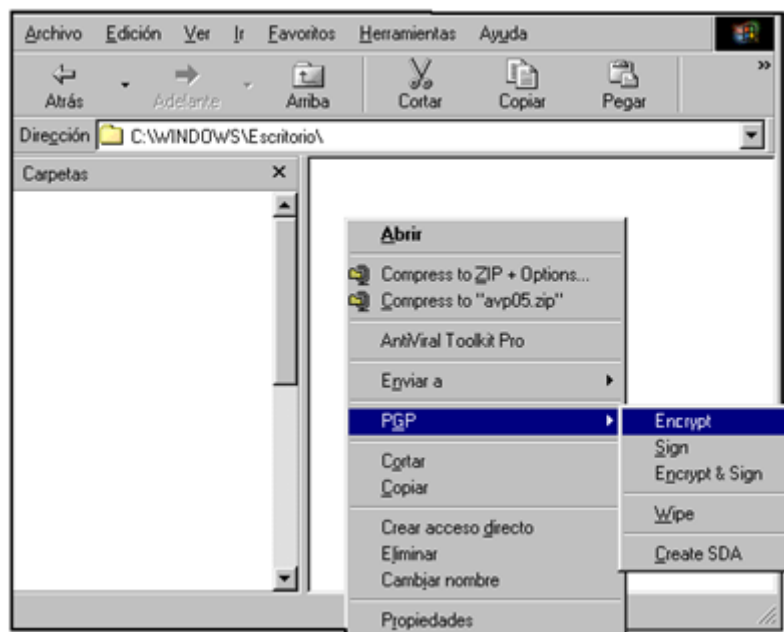


Figura 65. Funcionamiento para cifrar ficheros.

Una vez le seleccionamos la opción de cifrar, nos aparecerá un cuadro de dialogo para elegir ala forma y la clave publica del destinatario. Las opciones son las siguientes:

- ❖ **Text Output:** Esta opción solo es necesaria para enviar ficheros adjuntos a un correo.
- ❖ **Wipe Original:** Al marcar esta opción, cuando PGP genera el archivo cifrado, borra el original (de forma segura). Si no se marca, tendremos tanto el original como el cifrado.
- ❖ **Conventional Encryption:** Con esta opción lo que haremos es cifrar nuestro fichero. El fichero quedará protegido mediante una frase clave que nos pedirá que le indiquemos.
- ❖ **Self Description Archive:** Esta opción genera el fichero cifrado de forma que pueda ser descifrado directamente. Es muy útil para enviar ficheros a otras personas que no tiene PGP.



Para descifrar el fichero, sólo tenemos que pulsar dos veces sobre él y nos aparecerá el cuadro de dialogo con las mismas opciones que para cifrar pero en este caso para descifrar.

### 7.7.2. Funcionamiento con el programa de correo Outlook Express

El uso de PGP con el programa de correo Outlook Express, con la activación de la pestaña realizada anteriormente, al arrancar el cliente de correo se nos aparecerá unas nuevas opciones:

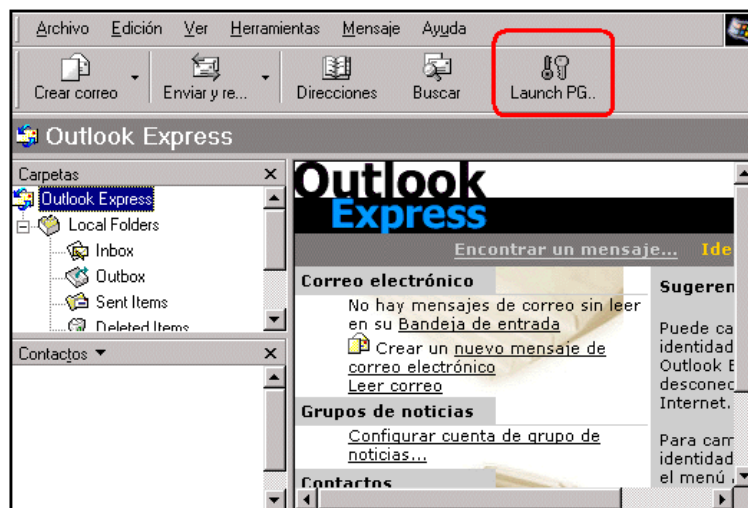


Figura 66. Funcionamiento con el programa de correo Outlook Express.

Lo primero que tenemos que hacer es crear un mensaje nuevo:

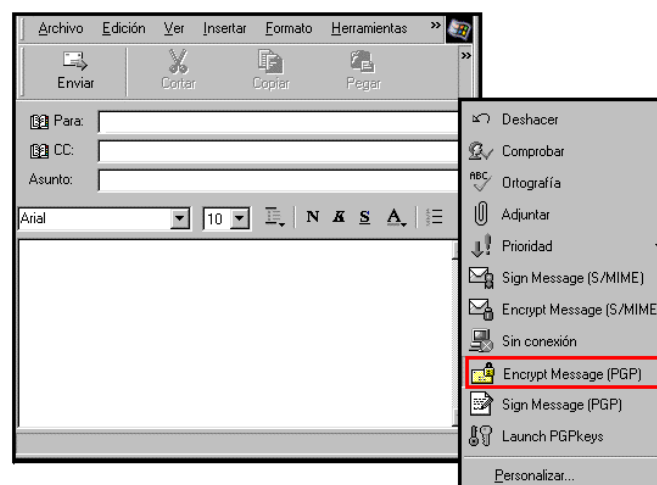
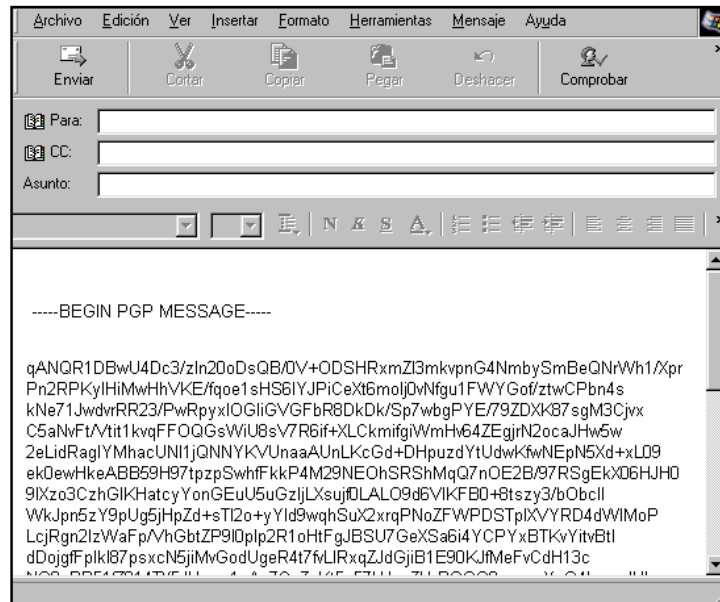


Figura 67. Crear mensaje nuevo.



Una vez que tenemos la opción de cifrar el mensaje, lo enviamos normalmente, pulsando **enviar**. Es cuando PGP se encarga de convertir el mensaje cifrado. Este mensaje quedaría como se muestra en la *Figura 68*.



**Figura 68. Mensaje cifrado.**



## **CAPITULO 8. Conclusiones**

La seguridad en las redes de comunicación se está convirtiendo en un factor importante en la actualidad, debido a la presencia de diferentes ataques e intrusos en la red, lo que nos lleva a la conclusión de que son circunstancias que afectan al funcionamiento de la red e incluso a la protección de nuestros datos. La principal consecuencia surge por el crecimiento tan elevado de usuarios en Internet, lo que se convierte en un medio indispensable para la comunicación.

Esto nos obliga a emplear cada vez más sistemas de seguridad que ofrezcan protección para mantener una total disponibilidad de los recursos y servicios en la red.

Con la elaboración de este proyecto fin de carrera se ha intentado transmitir y explicar algunos de los posibles métodos que hemos utilizado a la hora de proteger y mejorar la seguridad en las redes de comunicación, ya no sólo a nivel teórico sino también mediante una serie de explicaciones de las distintas herramientas utilizadas de cómo realizar diferentes ataques en una serie de escenarios con el fin de demostrar lo sencillo que resulta romper la seguridad de nuestras contraseñas.

Finalmente como conclusiones podemos decir que:

- ❖ La información tanto a nivel de usuario como de una organización es necesario protegerla contra las diferentes amenazas a la que quedan expuestas, mediante el análisis de las posibles vulnerabilidades, y principalmente minimizando los posibles riesgos.
- ❖ Existen diferentes herramientas de autenticación, autorización y cifrado de la información para mantener un buen funcionamiento de la red y protegerla ante los diferentes ataques.
- ❖ Los escáneres de red permite principalmente a una empresa especificar cuales de ellos son más probables que ocurran, aquellos que pueden ser mas peligrosos y cuáles de ellos son los más urgentes de minimizar.
- ❖ En las políticas de seguridad de la red podemos evaluar los riesgos existentes y los controles de acceso posibles.



## **BIBLIOGRAFÍA**

“Seguridad en Internet: Una guía practica y eficaz para proteger su Pc con software gratuito”, Gonzalo Asensio, Ed. Nowtilus S.L., 2006.

James F. Koruse, Keith W. Ros, “Redes de Computadores. Un enfoque descendente basado en Internet,” 2ª edición, Pearson Education, 2003. ISBN 84-7829-061-3. Capítulo 7.

William Stalling, “Fundamentos de seguridad en redes de aplicaciones y estándares,” 2ª edición, Pearson Education, 2003. ISBN 84-205-4002-1. Capítulo 5.

<http://anelkaos.googlepages.com/WiFiSlax2.0.pdf>

<http://www.imgeek.net/?p=88>

<http://tomcat.apache.org/>

[http://es.wikipedia.org/wiki/Apache\\_Tomcat](http://es.wikipedia.org/wiki/Apache_Tomcat)

<http://www.programacion.com/tutorial/tomcatintro/>

[http://www.elhacker.net/manual\\_hacking\\_wireless.htm](http://www.elhacker.net/manual_hacking_wireless.htm)

<http://axlinux.blogspot.com/2007/11/como-recuperar-las-contrasenas-de.html>

<http://gnu-linux-fsloizp.blogspot.com/2007/12/recuperar-contraseas-de-pdf-en-ubuntu.html?showComment=1220657580000>

<http://foro.portalhacker.net/index.php/topic,42476.0.html>

<http://www.insecure.org/nmap>

<http://www.cve.mitre.org>

<http://www.nessus.org/nessus>

<http://www.funk.com/download.asp?acg=6>

[www.freeradius.org](http://www.freeradius.org)

[www.ietf.org](http://www.ietf.org)

<http://www.modssl.org>

<http://www.apache.org>

<http://www.opnessl.org>

<http://www.modssl.org>

<https://www.verisign.com/products/srv/trial/intro.html>