Capítulo 8

Fase de pruebas

Introducción

Instalación del S.O. y del sistema base

Configuración de los paquetes necesarios

Pruebas de funcionamiento

Capítulo 8 Fase de pruebas

Tras la teoría desarrollada en los dos bloques anteriores, pasamos ahora a contemplar todos los pasos que hemos dado para configurar el sistema y que trabaje de la forma deseada.

Someteremos el sistema a una batería de pruebas para confirmar su buen funcionamiento, acorde al diseño y a la política inicial.

8.1 Introducción

Esta parte del proyecto consiste en una serie de pruebas que servirán para confirmar el correcto funcionamiento del sistema.

Los pasos a seguir vendrán determinados por cada paquete instalado en concreto. Se intentará probar la mayor parte de opciones de cada uno de ellos, aunque se dará un mayor seguimiento a las funcionalidades específicas del sistema. Esto es, aunque nuestro principal cometido es la autenticación, no vamos a validar un paquete cuyo funcionamiento no sea el adecuado.

En primer lugar se indicará cómo instalar y configurar el sistema, y a continuación, se mostrará el desarrollo de las pruebas realizadas.

8.2 Instalación y configuración

8.2.1 Instalación del sistema operativo

Para la instalación vamos a partir de un sistema limpio, que sólo tiene instalado la distribución Red Hat 9.0. Exactamente la hemos instalado en modo servidor, quitado algunos paquetes y añadidos otros.

En concreto no hemos instalado los paquetes que automáticamente se seleccionan en el modo servidor, y que mostramos a continuación:

- En el apartado Servidor:
  - Herramientas de configuración del servidor
  - Servidor web
  - Servidor de ficheros Windows

- En el apartado Sistema
  - Herramientas de administración
  - Soporte para la impresión

Los paquetes que sí hemos instalados y que no se encuentran en la instalación tipo servidor se encuentran en el apartado desarrollo. Para incluirlos sólo debemos activar siguiente opción:

- Apartado Desarrollo
  - Herramientas de desarrollo

Las herramientas de desarrollo son necesarias en cuanto que estamos compilando la mayor parte del software desde fuente. Hacen falta pues programas como gcc, make y demás.

Los paquetes que han sido eliminados no van a hacer falta, por lo que hemos optado por no instalarlos.

8.2.2 Instalación del sistema base

El sistema base lo componen los siguientes paquetes:

- OpenLDAP y sus dependencias:
  - Berkeley DB
  - Cyrus Sasl
  - Openssl
- Módulos pam_ldap y nss_ldap
- Ficheros de configuración

El sistema base comprende la configuración de lo que hemos denominado servicios Unix, para que se autentiquen contra el servidor de directorios. Mediante su correcto funcionamiento comprobaremos que el servidor se comporta tal como deseamos.

La instalación del sistema base está prácticamente automatizada mediante una serie de scripts. Al principio del cuerpo de cada uno de ellos, podemos encontrar una lista de variables, que se corresponden con los parámetros necesarios para la configuración del paquete en concreto.

La zona donde definimos estos parámetros es claramente distinguible para proceder a su personalización, aunque con los valores que aparecen por defecto es posible hacer una instalación de prueba.

Tras ejecutar cada uno de ellos, podemos consultar los archivos de logs para cerciorarnos de que todo el proceso se ha llevado a cabo satisfactoriamente.

Se usará el directorio /usr/local/src, inicialmente vacío, como raíz para contener todos los ficheros de los que hagamos uso, incluidos scripts.

Para comenzar la instalación debemos copiar los archivos necesarios al disco duro. Este proceso también lo hemos automatizado, por lo que la instalación se reduce a la ejecución de dos scripts.

Partimos de la estructura de directorio del cd que se incluye con la documentación.

```
mount /dev/cdrom /mnt
cd /mnt
./COPIAR_AL_DD.sh
```

Tras la ejecución del último comando ya disponemos de los archivos necesarios en el directorio /usr/local/src, como ya indicamos anteriormente. Sólo queda instalar.

```
cd /usr/local/src
./INSTALAR_BASE.sh
```

Este proceso es bastante largo, tarda en terminar en torno a 30 minutos (estimación realizada en el equipo de prueba). La cantidad de tiempo que lleva es normal, puesto que todos los paquetes, compuestos por multitud de archivos fuente, han de ser compilados.

Ya instalado, el siguiente paso sería poblar e iniciar el servidor, para así poder comprobar su estado.

```
./poblar.sh
```

No debe dar ningún error.

8.2.3 Instalación de squid

La instalación del proxy - caché se realiza fácilmente mediante el script creado para tal efecto. Se encuentra en el mismo directorio que se toma como raíz durante toda la instalación.

El script también realiza los cambios necesarios en los ficheros de configuración y de arranque. Su ejecución se realiza de esta forma:

```
./INSTALAR_SQUID.sh
```

8.2.4 Instalación de postgres

De la misma forma que Squid, el gestor de bases de datos postgres, dispone de un script que facilita la instalación. De nuevo, también se encarga de los ficheros de configuración y de arranque.

```
./INSTALAR_POSTGRES.sh
```

8.2.5 Instalación de qmail

Durante el capítulo dedicado a qmail, ya hicimos uso de diferentes scripts para llevar a cabo su instalación. Ahora se usará uno más general, que contempla a los anteriores, de forma que se automaticen todos los pasos.

Tras la instalación de qmail propiamente dicha, se efectuará la misma operación sobre SqWebMail y sus dependencias. La separación en dos partes tiene un fin meramente aclarativo.

```
./INSTALAR_QMAIL.sh
./INSTALAR_SQWEBMAIL.sh
```

8.2.6 Instalación de los certificados digitales

La creación de certificados y claves no ha sido automatizada ya que requiere una participación bastante interactiva por parte del administrador.

Existen detalles, tales como tiempo de expiración o descripción, que deben quedar bajo el criterio de esta persona en cuestión. Incluso si el campo FQDN del servidor no coincide con su nombre real, el certificado no tendría validez, y por tanto no lo usaría el servidor.

Vamos a necesitar los certificados digitales en dos ocasiones.

- Para autenticar al servidor OpenLDAP y negociar una capa segura (TLS) entre clientes y el mismo servidor. Protegemos así el protocolo LDAP.
- Para autenticar al servidor apache y negociar una capa segura (SSL) entre cliente y el mismo servidor. Protegemos así el protocolo HTTP.

Toda la información acerca de los certificados digitales y su uso la podemos encontrar el el apéndice. De todas formas hemos incluido varios scripts para ayudarnos en esta labor.

Éstos se encuentran en el directorio certificados, que a su vez se encuentra en el directorio raíz de nuestra instalación, /usr/local/src. Además también se adjuntan los archivos necesarios que deben ser copiados a los clientes y un fichero de ayuda con sencillas instrucciones.

Debemos ejecutarlos y responder a sus preguntas convenientemente, tal como se indica en el apéndice.

```
cd /usr/local/src/certificados
./certificados_ldap.sh
./certificados_http.sh
```

8.3 Pruebas de funcionamiento

8.3.1 Descripción del entorno de pruebas

Para la realización de las pruebas hemos contado con una serie de equipos que citamos a continuación:

- Equipo "gotche", CPU de 2000 MHz y 512 MB de RAM, con S.O. Red Hat.
- Equipo "fred", CPU de 450 MHz y 64 MB de RAM, con S.O. Debian (Sarge).
- Equipo "kinko", CPU de 2000 MHz y 512 MB de RAM, con S.O. Windows.
- Router neutro USR Robotics de cuatro puertos
- Cablemodem Motorola para la conexión con internet.

La red está dispuesta de esta forma.

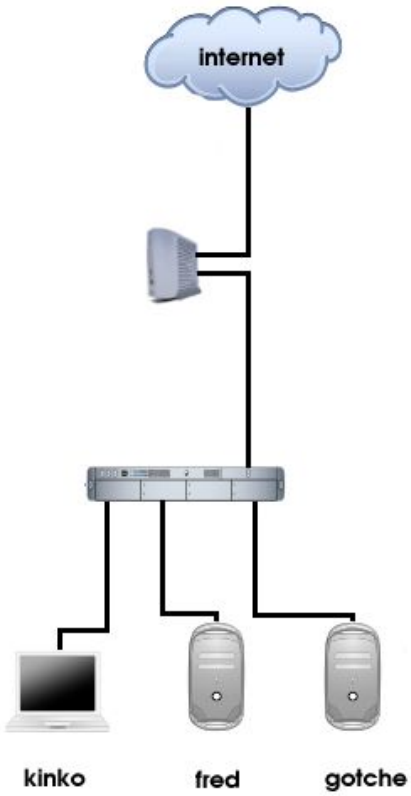

Figura 8.1 Red local donde se realizaron las pruebas

Gracias al router tenemos una subred con direcciones de tipo C. Éstas son:

gotche	192.168.123.100
kinko	192.168.123.101
fred	192.168.123.102
router (de cara a la red)	192.168.123.254
router (de cara al exterior)	dinámica

Los equipos cogen dirección por DHCP. Es el router el que dispone de un servidor DHCP, aunque está configurado para que sean fijas, de tal modo que no cambien dependiendo de qué equipo arranca antes.

La IP pública que nos proporciona nuestro ISP es dinámica. Para ofrecer un servicio, como el correo electrónico, por ejemplo, necesitamos un nombre de dominio.

Hemos resuelto este problema gracias a un servicio gratuito que nos ofrece la web www.dyndns.org. En ella podemos registrar un nombre de dominio con la dirección IP que deseemos. Para actualizar la dirección cuando ésta cambie, puesto que como ya hemos dicho es dinámica, utilizamos un programa cliente. En nuestro caso hemos usado ddclient, un programa también gratuito.

El dominio registrado ha sido gotche.dyndns.org, haciendo alusión a la máquina gotche, que es el que ejecuta ddclient cada vez que arranca, y por tanto gestiona los cambios de IP.

El equipo gotche, con S.O. Red Hat, albergará a los servidores OpenLDAP, Squid, Postgres y qmail. Será por ello nuestro punto de atención e incluso haremos pruebas de tipo cliente con él.

El equipo fred nos permitirá realizar pruebas de tipo cliente. Hemos elegido que use Debian por diversos motivos. Uno de ellos era por cambiar de sistema operativo (sin abandonar Linux, claro), aunque el fundamental es la facilidad para instalar programas, gracias a la utilidad apt-get.

Por último también disponemos de un equipo con S.O. Windows, con el que podremos realizar una serie de pruebas. No podremos probar todos los servicios (por ejemplo, su), pero si algunos.

8.4 Probando OpenLDAP

Para determinar el correcto funcionamiento del servidor OpenLDAP vamos a definir una batería de pruebas, ejecutarla desde cada equipo implicado, si procede, y comprobar sus resultados.

Tras la fase de compilación tenemos la oportunidad de ejecutar una serie de tests. Esto se realiza mediante la instrucción make tests ya comentada en el capítulo dedicado a OpenLDAP. Podemos comprobar el resultado de dichos tests inspeccionando los ficheros de logs generados en el momento de la compilación.

El resultado exitoso de las pruebas anteriores unido a que todo el funcionamiento del sistema descansa sobre el servidor, hará que nuestra batería de pruebas sea pequeña y sencilla. Vamos a poder ver el funcionamiento del servidor durante todas las pruebas siguientes así que no se va a alargar demasiado esta parte.

Pruebas.

1. Iniciar el servidor

   ```
   /etc/init.d/slapd start
   ```

   Resultado

   ```
   Iniciando OpenLDAP                [OK]
   ```

2. Reiniciar el servidor

   ```
   /etc/init.d/slapd restart
   ```

   Resultado:

   ```
   Apagando OpenLDAP                 [OK]
   Iniciando OpenLDAP                [OK]
   ```

3. Apagar el servidor

   ```
   /etc/init.d/slapd stop
   ```

   Resultado:

   ```
   Apagando OpenLDAP                 [OK]
   ```

4. Búsqueda con el servidor apagado

   ```
   ldapsearch –x uid=jose
   ```

Resultado:

```
ldap_bind: Can't contact LDAP server (-1)
```

5. Búsqueda con el servidor iniciado

```
ldapsearch -x uid=jose
```

Resultado:

```
# extended LDIF
#
# LDAPv3
# base <> with scope sub
# filter: uid=jose
# requesting: ALL
#

# jose, People, example.com
dn: uid=jose,ou=People,dc=example,dc=com
uid: jose
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
shadowLastChange: 12702
shadowMin: -1
shadowMax: 99999
shadowWarning: -1
loginShell: /bin/bash
uidNumber: 500
gidNumber: 500
homeDirectory: /home/jose

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

6. Añadir entradas con el servidor apagado (ideal cuando estamos efectuando una precarga)

```
/usr/local/bin/slapadd -l base.ldif
```

Resultado:

(si todo es correcto, el comando anterior no devuelve nada)

7. Añadir entradas con el servidor arrancado

```
/usr/local/bin/ldapadd -f pass.ldif -D "cn=Manager,dc=example,dc=com"
-w secret -x
```

donde el fichero pass.ldif contiene entradas de este tipo:

```
...
dn: uid=usuarioz,ou=People,dc=example,dc=com
uid: usuarioz
cn: zeta
objectClass: account
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}$1$RG4NnjLV$.bTsu0IVWHwrrl.Rzb3Pt.
ShadowLastChange: 12703
shadowMin: -1
shadowMax: 99999
shadowWarning: -1
loginShell: /bin/bash
uidNumber: 520
gidNumber: 520
homeDirectory: /home/usuarioz
gecos: usuario z
...
```

Resultado

```
adding new entry "uid=root,ou=People,dc=example,dc=com"
adding new entry "uid=bin,ou=People,dc=example,dc=com"
adding new entry "uid=daemon,ou=People,dc=example,dc=com"
adding new entry "uid=adm,ou=People,dc=example,dc=com"
adding new entry "uid=lp,ou=People,dc=example,dc=com"
adding new entry "uid=sync,ou=People,dc=example,dc=com"
adding new entry "uid=shutdown,ou=People,dc=example,dc=com"
adding new entry "uid=halt,ou=People,dc=example,dc=com"
adding new entry "uid=mail,ou=People,dc=example,dc=com"
adding new entry "uid=news,ou=People,dc=example,dc=com"
adding new entry "uid=uucp,ou=People,dc=example,dc=com"
adding new entry "uid=operator,ou=People,dc=example,dc=com"
adding new entry "uid=games,ou=People,dc=example,dc=com"
adding new entry "uid=gopher,ou=People,dc=example,dc=com"
adding new entry "uid=ftp,ou=People,dc=example,dc=com"
adding new entry "uid=nobody,ou=People,dc=example,dc=com"
adding new entry "uid=rpm,ou=People,dc=example,dc=com"
adding new entry "uid=vcsa,ou=People,dc=example,dc=com"
adding new entry "uid=nscd,ou=People,dc=example,dc=com"
```

```
adding new entry "uid=sshd,ou=People,dc=example,dc=com"
adding new entry "uid=rpc,ou=People,dc=example,dc=com"
adding new entry "uid=rpcuser,ou=People,dc=example,dc=com"
adding new entry "uid=nfsnobody,ou=People,dc=example,dc=com"
adding new entry "uid=mailnull,ou=People,dc=example,dc=com"
adding new entry "uid=smmsp,ou=People,dc=example,dc=com"
adding new entry "uid=pcap,ou=People,dc=example,dc=com"
adding new entry "uid=apache,ou=People,dc=example,dc=com"
adding new entry "uid=postgres,ou=People,dc=example,dc=com"
adding new entry "uid=squid,ou=People,dc=example,dc=com"
adding new entry "uid=alias,ou=People,dc=example,dc=com"
adding new entry "uid=qmaild,ou=People,dc=example,dc=com"
adding new entry "uid=qmaill,ou=People,dc=example,dc=com"
adding new entry "uid=qmailp,ou=People,dc=example,dc=com"
adding new entry "uid=qmailq,ou=People,dc=example,dc=com"
adding new entry "uid=qmailr,ou=People,dc=example,dc=com"
adding new entry "uid=qmails,ou=People,dc=example,dc=com"
adding new entry "uid=jose,ou=People,dc=example,dc=com"
adding new entry "uid=espe,ou=People,dc=example,dc=com"
adding new entry "uid=usuariox,ou=People,dc=example,dc=com"
adding new entry "uid=usuarioy,ou=People,dc=example,dc=com"
```

8. Borrar entradas

```
/usr/local/bin/ldapadd -f borrado.ldif -D
"cn=Manager,dc=example,dc=com" -w secret -x
```

donde el contenido del fichero borrado.ldif es el siguiente:

```
dn: uid=usuarioz,ou=People,dc=example,dc=com
changetype: delete
```

Resultado:

```
deleting entry "uid=usuarioz,ou=People,dc=example,dc=com"
```

9. Modificar entradas

```
/usr/local/bin/ldapadd -f cambio.ldif -D
"cn=Manager,dc=example,dc=com" -w secret -x
```

donde el contenido del fichero cambio.ldif es el siguiente:

```
dn: uid=usuarioz,ou=People,dc=example,dc=com
changetype: modify
replace: gecos
gecos: el usuario zeta vive en Cordoba
```

Resultado:

```
modifying entry "uid=usuarioz,ou=People,dc=example,dc=com"
```

Ejecución de las pruebas desde los distintos equipos

Nº de prueba	*Equipo gotche*	*Equipo fred*	*Equipo kinko*
1	✓	✓	(no procede)
2	✓	✓	(no procede)
3	✓	✓	(no procede)
4	✓	✓	(no procede)
5	✓	✓	(no procede)
6	✓	✓	(no procede)
7	✓	✓	(no procede)
8	✓	✓	(no procede)
9	✓	✓	(no procede)

Para realizar las pruebas desde el equipo con Debian, fred, se necesitó instalar las librerías y clientes de LDAP correspondientes. Se trata de los paquetes ldap-utils y libldap2.

En cuanto al equipo en Windows, kinko, no hemos realizado ninguna prueba ya que no existe la necesidad de este uso.

8.5 Probando servicios Unix

Para realizar estas pruebas hemos contado con una serie de usuarios. En particular vamos a hacer uso de dos, usuariox y usuarioy. El primero de ellos se encuentra tanto en el fichero /etc/passwd como en el directorio. Usuarioy solamente se encuentra en el directorio.

De esta forma tan simple vamos a comprobar que los servicios Unix utilizan el directorio y no el fichero de password. Además compararemos las salidas que produce el usuario normal y el integrado y veremos que son idénticas.

Pruebas

1. Su. Cambio a usuariox desde root.
   `su usuariox`
   Resultado
   `[usuariox@gotche usuariox]$` (cambio en el prompt, red hat)
   `usuariox@fred :~$` (cambio en el prompt, debian)
2. Su. Cambio a usuarioy desde root.
   `su usuarioy`
   Resultado
   `[usuarioy@gotche usuarioy]$` (cambio en el prompt, red hat)
   `usuarioy@fred :~$` (cambio en el prompt, debian)
3. Su. Cambio de usuariox a usuarioy
   `su usuarioy`
   Resultado:
   `Password:`
   `[usuarioy@gotche usuarioy]$` (cambio en el prompt, red hat)

   `Password:`
   `usuarioy@fred :~$` (cambio en el prompt, debian)
4. Su. Cambio de usuarioy a usuariox
   `su usuariox`
   Resultado:

```
Password:
[usuariox@gotche usuariox]$ (cambio en el prompt, red hat)

Password:
usuariox@fred :~$           (cambio en el prompt, debian)
```

5. Login. Inicio de sesión del usuariox

```
gotche login: usuariox      (red hat)
fred login: usuariox        (debian)
```

Resultado:

```
Password:
[usuariox@gotche usuariox]$ (cambio en el prompt, red hat)

Password:
usuariox@fred :~$           (cambio en el prompt, debian)
```

6. Login. Inicio de sesión del usuarioy

```
gotche login: usuarioy      (red hat)
fred login: usuarioy        (debian)
```

Resultado:

```
Password:
[usuarioy@gotche usuarioy]$ (cambio en el prompt, red hat)

Password:
usuarioy@fred :~$           (cambio en el prompt, debian)
```

7. Gdm. Inicio de sesión gráfica del usuariox

Tras arrancar los equipos gotche y fred aparecen sendos login gráficos.

Figura 8.2 Login gráfico para la distribución Red Hat

Figura 8.3 Login gráfico para la distribución Debian

Resultado:

Entra en el escritorio

8. Gdm. Inicio de sesión gráfica del usuarioy

   Es la misma prueba que la anterior. Aparecen las figuras 8.2 y 8.3.

   Resultado:

   Entra en el escritorio

9. Ssh. Sesión ssh mediante usuariox

   ```
   ssh usuariox@gotche
   ```

   Resultado:

   ```
   usuariox@gotche's password:
   [usuariox@gotche usuariox]$
   ```

10. Ssh. Sesión ssh mediante usuarioy

    ```
    ssh usuarioy@gotche
    ```

    Resultado:

    ```
    usuarioy@gotche's password:
    [usuarioy@gotche usuarioy]$
    ```

11. Ssh. Sesión ssh mediante usuariox (introduciendo una contraseña falsa)

    ```
    ssh usuariox@gotche
    ```

    Resultado:

    ```
    usuariox@gotche's password:
    Permission denied, please try again.
    usuariox@gotche's password:
    ```

12.Ssh. Sesión ssh mediante usuarioy (introduciendo una contraseña falsa)

```
ssh usuarioy@gotche
```

Resultado:

```
usuarioy@gotche's password:
Permission denied, please try again.
usuarioy@gotche's password:
```

13.Ssh. Sesión sftp mediante usuariox

```
sftp usuariox@gotche
```

Resultado:

```
Connecting to gotche...
usuariox@gotche's password:
sftp>
```

14.Ssh. Sesión sftp mediante usuarioy

```
sftp usuarioy@gotche
```

Resultado:

```
Connecting to gotche...
usuarioy@gotche's password:
sftp>
```

15.Ftp. Sesión ftp mediante usuariox

```
ftp gotche
Connected to gotche.
220 (vsFTPd 1.1.3)
Name (gotche:root):             [Pulsamos enter]
530 Permission denied.
Login failed.
ftp> user usuariox              [Introducimos el usuario]
```

Resultado:

```
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is Unix.
Using binary mode to transfer files.
ftp>
```

16.Ftp. Sesión ftp mediante usuarioy

```
ftp gotche
Connected to gotche.
220 (vsFTPd 1.1.3)
Name (gotche:root):             [Pulsamos enter]
530 Permission denied.
```

```
Login failed.
ftp> user usuarioy              [Introducimos el usuario]
```

Resultado:

```
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is Unix.
Using binary mode to transfer files.
ftp>
```

17.Passwd. Cambio de password del usuariox

```
passwd   [desde la cuenta del usuariox]
```

Resultado:

```
Changing password for user usuariox.
Enter login(LDAP) password:
New password:
Re-enter new password:
LDAP password information changed for usuariox
[usuariox@gotche usuariox]$
```

18.Passwd. Cambio de password del usuarioy

```
passwd   [desde la cuenta del usuarioy]
```

Resultado:

```
Changing password for user usuarioy.
Enter login(LDAP) password:
New password:
Re-enter new password:
LDAP password information changed for usuariox
[usuarioy@gotche usuarioy]$
```

19.Passwd. Cambio de password del usuariox (contraseña falsa)

```
passwd   [desde la cuenta del usuariox]
```

Resultado:

```
Changing password for user usuariox.
Enter login(LDAP) password:
LDAP password incorrect: try again
Enter login(LDAP) password:
```

20.Passwd. Cambio de password del usuarioy (contraseña falsa)

```
passwd   [desde la cuenta del usuarioy]
```

Resultado:

```
Changing password for user usuarioy.
```

```
Enter login(LDAP) password:
LDAP password incorrect: try again
Enter login(LDAP) password:
```

21. Halt,poweroff,shutdown. Apagado del equipo lanzado por usuariox

```
/sbin/halt
/sbin/poweroff
/sbin/shutdown -h now
```

Resultado:

```
halt: must be superuser
poweroff: must be superuser
shutdown: you must be root to do that!
```

22. Halt,poweroff,shutdown. Apagado del equipo lanzado por usuarioy

```
/sbin/halt
/sbin/poweroff
/sbin/shutdown -h now
```

Resultado:

```
halt: must be superuser
poweroff: must be superuser
shutdown: you must be root to do that!
```

23. Halt,poweroff,shutdown. Apagado del equipo lanzado por root

```
/sbin/halt
/sbin/poweroff
/sbin/shutdown -h now
```

Resultado:

(se apaga el equipo)

Ejecución de las pruebas desde los distintos equipos

Nº de prueba	*Equipo gotche*	*Equipo fred*	*Equipo kinko*
1	✓	✓	(no procede)
2	✓	✓	(no procede)
3	✓	✓	(no procede)
4	✓	✓	(no procede)

Nº de prueba	***Equipo gotche***	***Equipo fred***	***Equipo kinko***
5	✓	✓	(no procede)
6	✓	✓	(no procede)
7	✓	✓	(no procede)
8	✓	✓	(no procede)
9	✓	✓	✓
10	✓	✓	✓
11	✓	✓	✓
12	✓	✓	✓
13	✓	✓	(no procede)
14	✓	✓	(no procede)
15	✓	✓	✓
16	✓	✓	✓
17	✓	✓	(no procede)
18	✓	✓	(no procede)
19	✓	✓	(no procede)
20	✓	✓	(no procede)
21	✓	✓	(no procede)
22	✓	✓	(no procede)
23	✓	✓	(no procede)

8.6 Probando Squid

El funcionamiento de la autenticación en Squid será el siguiente: al abrir el navegador e intentar acceder a una página web por primera vez, aparecerá una ventana pidiendo nombre de usuario y contraseña. Si todo se introduce correctamente, esta ventana desaparece y se permite ver el contenido de la página web solicitada.

Durante todo el tiempo que el navegador esté abierto no se volverá a pedir clave de acceso, aunque el resto de reglas (filtros y demás) seguirán vigentes.

Para hacer uso del proxy hay que configurar el navegador. Debemos modificar los ajustes de conexión e indicar la dirección IP del proxy. El puerto que usa Squid es el 3128.

Para probar Squid hemos contado con diversos navegadores, entre ellos los más usados, Firefox y Explorer. Se han realizado capturas que muestran las pantallas de autenticación en cada uno de ellos.

Pruebas

1. Acceso a www.google por parte de usuariox

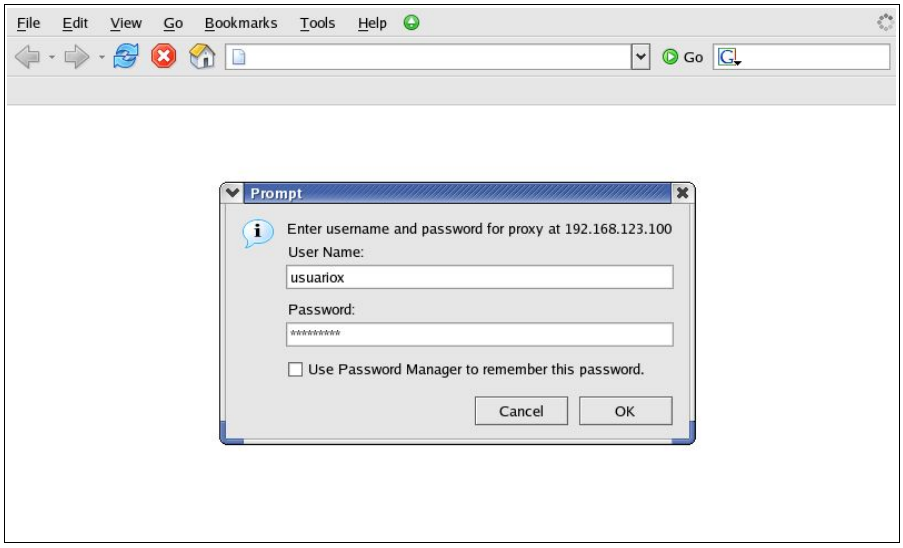

Figura 8.4 Acceso de usuariox a internet

Resultado:

Entra en la web

2. Acceso a www.google por parte de usuarioy

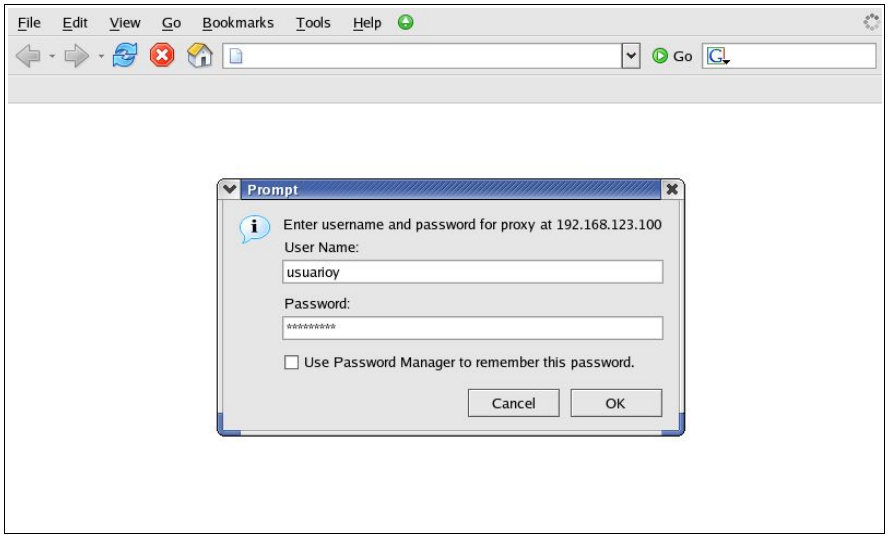

Figura 8.5 Acceso de usuariox a internet

Resultado:

Accede perfectamente.

3. Acceso a google por parte de usuariox (contraseña incorrecta)

   Resultado:

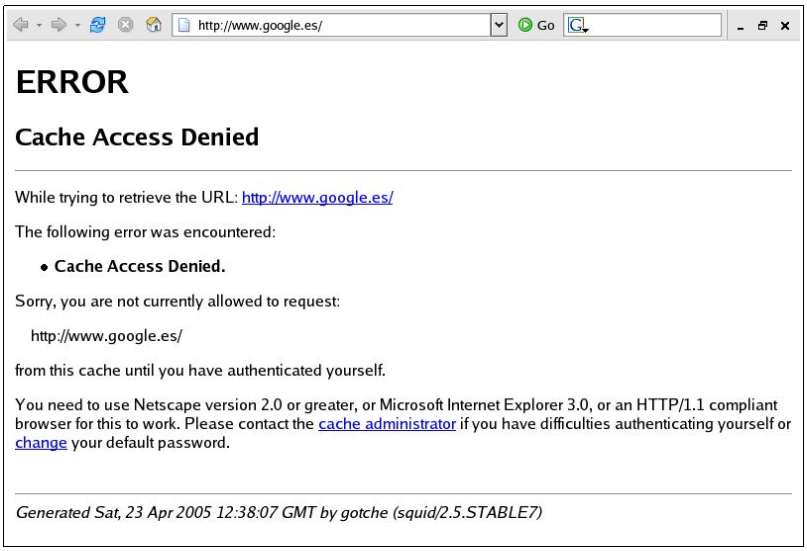

Figura 8.6 Página de denegación de acceso de Squid

4. Acceso a google por parte de usuarioy (contraseña incorrecta)

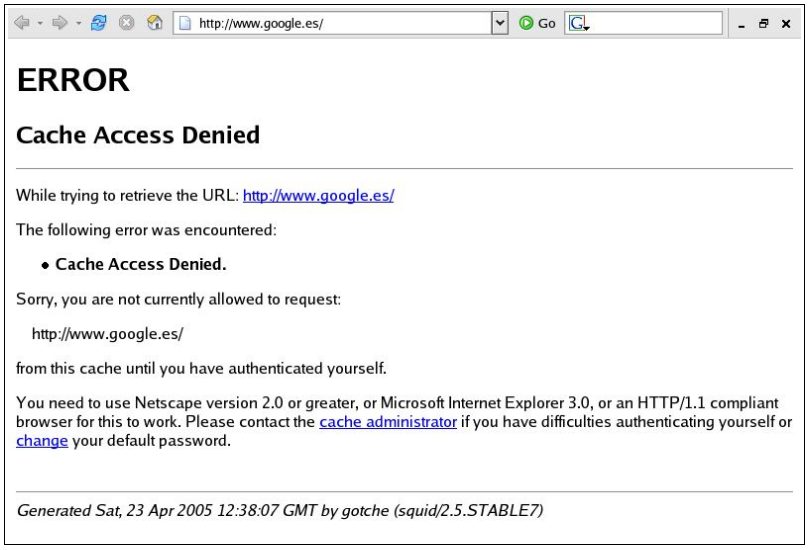

Figura 8.7 Página de denegación de acceso de Squid

5. Acceso a una página que no existe.

Figura 8.8 Página de denegación de acceso de Squid

6. Acceso a una página que está filtrada.

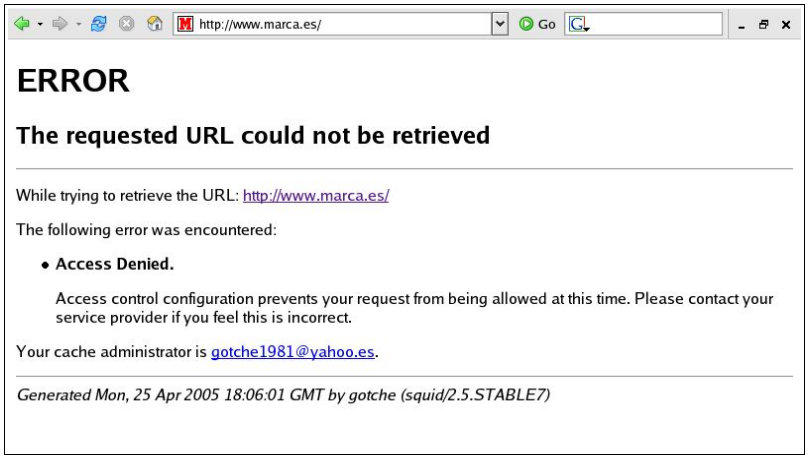

Figura 8.9 Acceso denegado a una página filtrada

Ejecución de las pruebas desde los distintos equipos

Nº de pruebas	Equipo gotche	Equipo fred	Equipo kinko
1	✓	✓	✓
2	✓	✓	✓
3	✓	✓	✓
4	✓	✓	✓
5	✓	✓	✓
6	✓	✓	✓

8.7 Probando Postgres

La gestión de los usuarios en Postgres difiere un poco de lo que hemos venido realizando hasta ahora. Para que un usuario pueda acceder a una base de datos no sólo debe tener una cuenta en el sistema dónde se encuentre, sino que también debe tenerla dentro de Postgres.

Hemos aceptado esta política ya que es probable (e incluso conveniente) que no todos los usuarios del sistema tengan acceso a las bases de datos. No es muy usual que un usuario típico pueda acceder ni crear bases de datos, de ahí este tratamiento especial.

El tratamiento especial consiste en añadir el usuario mediante una herramienta de Postgres. Hecho esto, para acceder a una base de datos basta suministrar la contraseña del usuario, la del directorio. Tenemos así, la autenticación de Postgres integrada.

Por último indicar que para probar postgres, desde el equipo con S.O. Debian, se ha tenido que instalar en él los paquetes necesarios de cliente de postgres.

Pruebas

1. Creación de una base de datos de prueba (llamada test) por parte del usuario postgres

   ```
   /usr/local/pgsql/bin/createdb test
   ```

   Resultado:

   ```
   Password:
   CREATE DATABASE
   [postgres@gotche postgres]$
   ```

2. Acceso del usuario postgres

   ```
   /usr/local/pgsql/bin/psql test –U postgres
   ```

   Resultado:

   ```
   Password:
   Welcome to psql 7.4.5, the PostgreSQL interactive terminal
   ```

```
Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit

test=#
```

3. Acceso del usuariox

```
/usr/local/pgsql/bin/psql test -U usuariox
```

Resultado:

```
Password:
psql: FATAL:  user "usuariox" does not exist
```

4. Creación del usuariox dentro de Postgres

```
/usr/local/pgsql/bin/createuser usuariox
```

Resultado:

```
Shall the new user be allowed to create databases? (y/n) n
Shall the new user be allowed to create more new users? (y/n) n
Password:
CREATE USER
```

5. Acceso del usuariox

```
/usr/local/pgsql/bin/psql test -U usuariox
```

Resultado:

```
Password:
Welcome to psql 7.4.5, the PostgreSQL interactive terminal

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit

test=#
```

6. Acceso del usuariox (contraseña incorrecta)

```
/usr/local/pgsql/bin/psql test -U usuariox
```

Resultado:

```
Password:
psql: FATAL:  PAM authentication failed for user "usuariox"
```

7. Acceso del usuarioy

```
/usr/local/pgsql/bin/psql test -U usuarioy
```

Resultado:

```
Password:
psql: FATAL:  user "usuarioy" does not exist
```

8. Creación del usuarioy

```
/usr/local/pgsql/bin/createuser usuarioy
```

Resultado:

```
Shall the new user be allowed to create databases? (y/n) n
Shall the new user be allowed to create more new users? (y/n) n
Password:
CREATE USER
```

9. Acceso del usuarioy

```
/usr/local/pgsql/bin/psql test -U usuarioy
```

Resultado:

```
Password:
Welcome to psql 7.4.5, the PostgreSQL interactive terminal

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help on internal slash commands
      \g or terminate with semicolon to execute query
      \q to quit

test=#
```

10. Acceso del usuarioy (contraseña incorrecta)

```
/usr/local/pgsql/bin/psql test -U usuarioy
```

Resultado:

```
Password:
psql: FATAL:  PAM authentication failed for user "usuarioy"
```

11. Carga de datos

```
test=# \i basico.sql
```

Resultado:

```
CREATE TABLE
CREATE TABLE
INSERT 17148 1
INSERT 17149 1
INSERT 17150 1
INSERT 17151 1
```

12.Consulta

```
test=# Select * FROM tiempo;
```

Resultado

```
  ciudad  | temp_baja | temp_alta | prcp |   fecha
---------+-----------+-----------+------+------------
 Sevilla |        12 |        25 | 0.25 | 2005-03-16
 Sevilla |        11 |        23 |    0 | 2005-03-14
 Cadiz   |        12 |        20 |      | 2005-03-16
(3 rows)
```

13.Borrado de tablas

```
test=# DROP TABLE tiempo, ciudades;
```

Resultado

```
DROP TABLE
test=#
```

Ejecución de las pruebas desde los distintos equipos

Nº de prueba	Equipo gotche	Equipo fred	Equipo kinko
1	✓	✓	(no procede)
2	✓	✓	(no procede)
3	✓	✓	(no procede)
4	✓	✓	(no procede)
5	✓	✓	(no procede)
6	✓	✓	(no procede)
7	✓	✓	(no procede)
8	✓	✓	(no procede)
9	✓	✓	(no procede)
10	✓	✓	(no procede)
11	✓	✓	(no procede)
12	✓	✓	(no procede)
13	✓	✓	(no procede)

8.8 Probando qmail y SqWebMail

En este apartado debemos mostrar el correcto funcionamiento de qmail y SqWebMail. Para ello podemos hacerlo por pasos. Primero veríamos qmail, emitiendo correos mediante telnet al puerto 25, analizando los archivos de logs creados y viendo los mensajes recibidos en los directorios Maildir.

Después veríamos SqWebMail. Esto sería algo más ameno y cercano, una aplicación gráfica, que es lo que estamos acostumbrados a manejar para trabajar con el correo.

Puesto que comprobando que SqWebMail permite mandar y recibir correo, estamos implícitamente diciendo lo mismo de qmail, vamos a trabajar directamente con el primero.

Pruebas

1. Acceso de usuariox

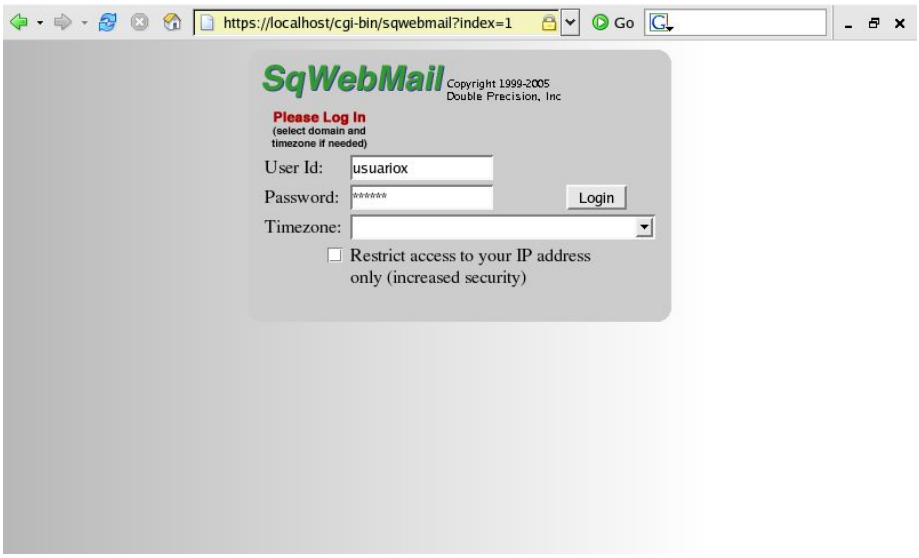

Figura 8.10 Acceso a SqWebMail

Resultado:

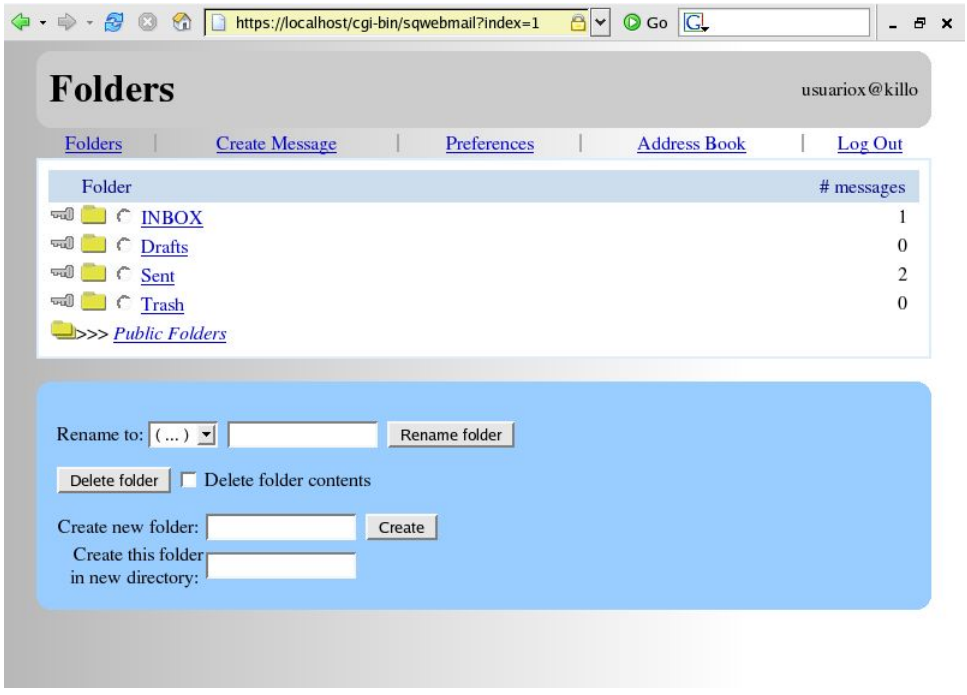

Figura 8.11 Usuariox entra en su cuenta de correo

2. Acceso de usuarioy

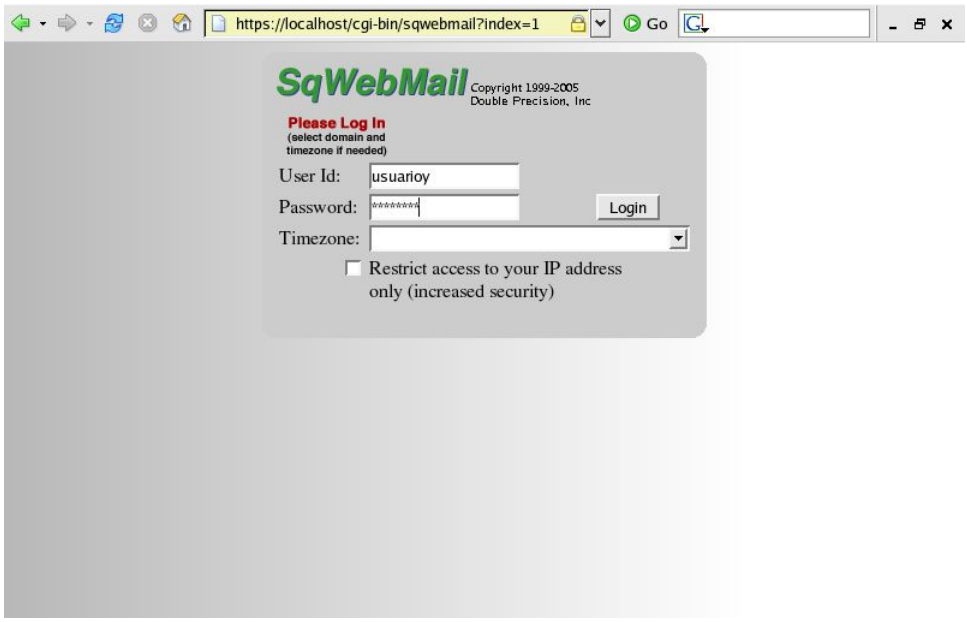

Figura 8.12 Acceso de usuarioy

Resultado:

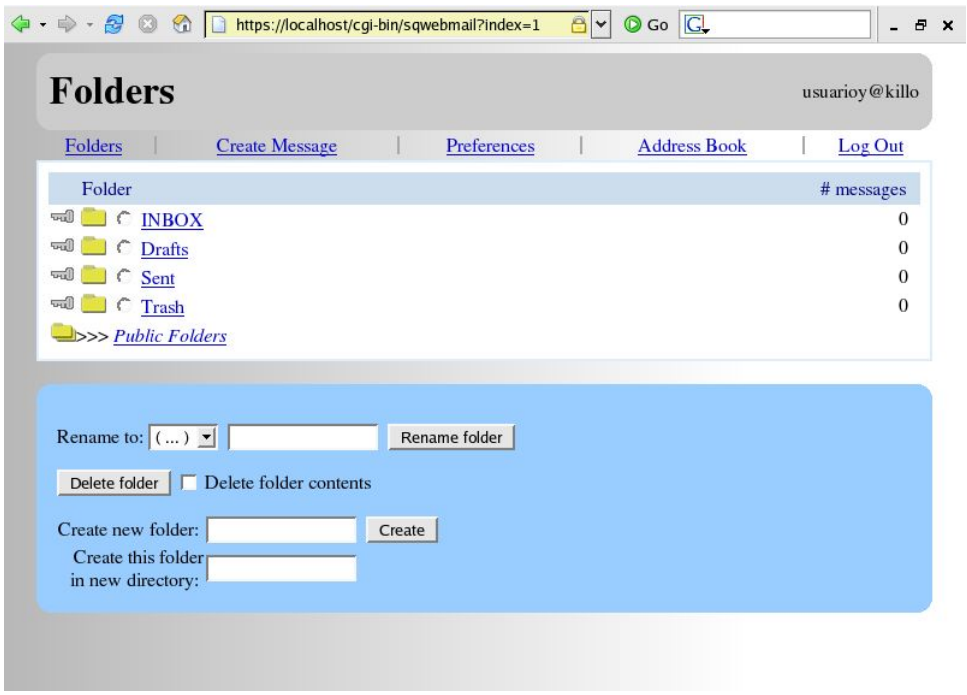

Figura 8.13 Usuariox entra en su cuenta de correo

3. Mandar correo de usuariox a un usuario de gmail (correo externo)

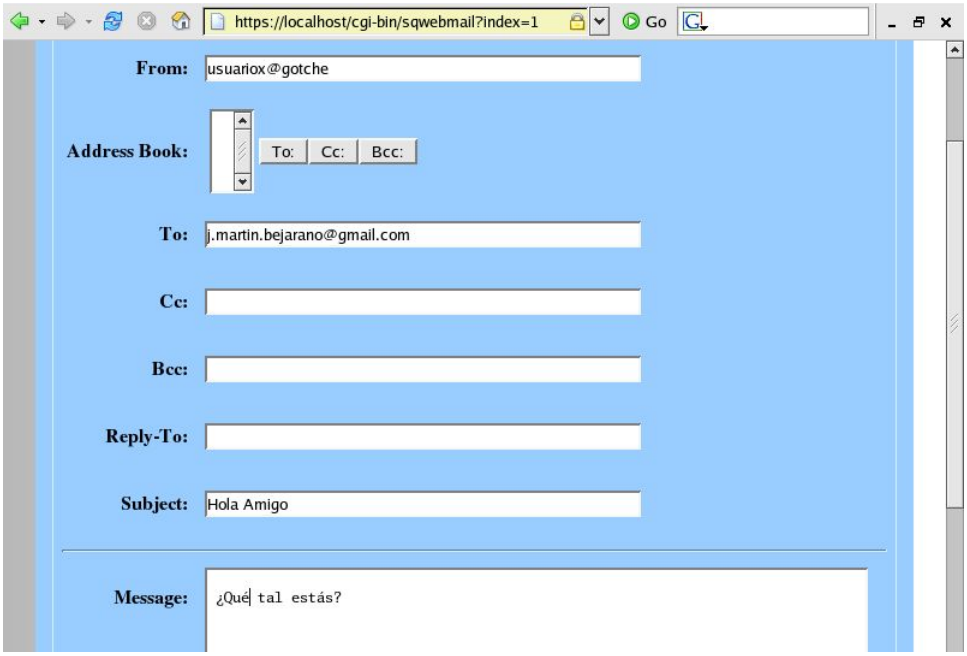

Figura 8.14 Usuariox envía un correo externo

Resultado:

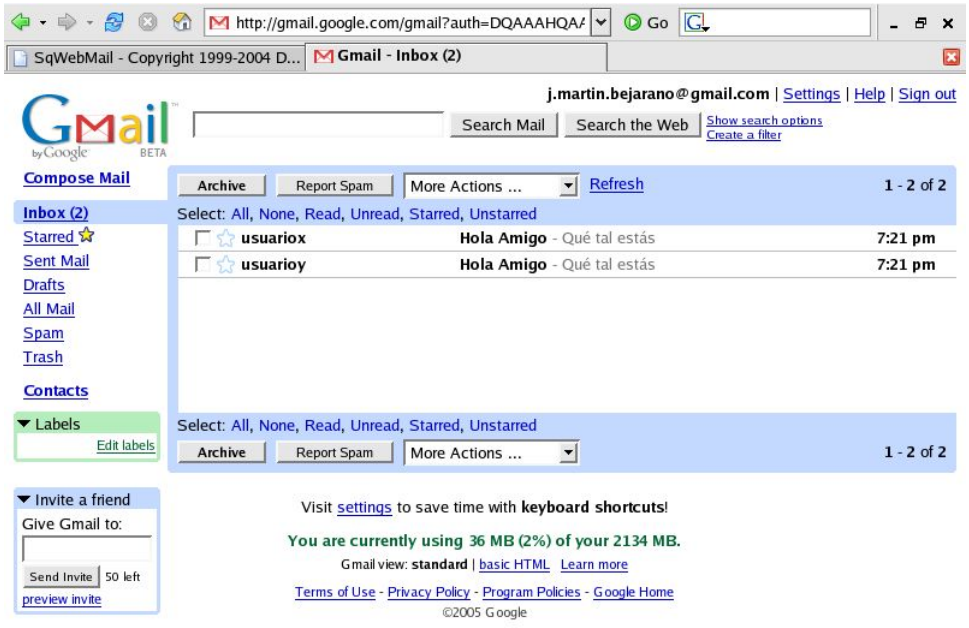

Figura 8.15 Recepción del mensaje por parte del usuario externo

4. Mandar correo de usuarioy a un usuario de gmail (correo externo)

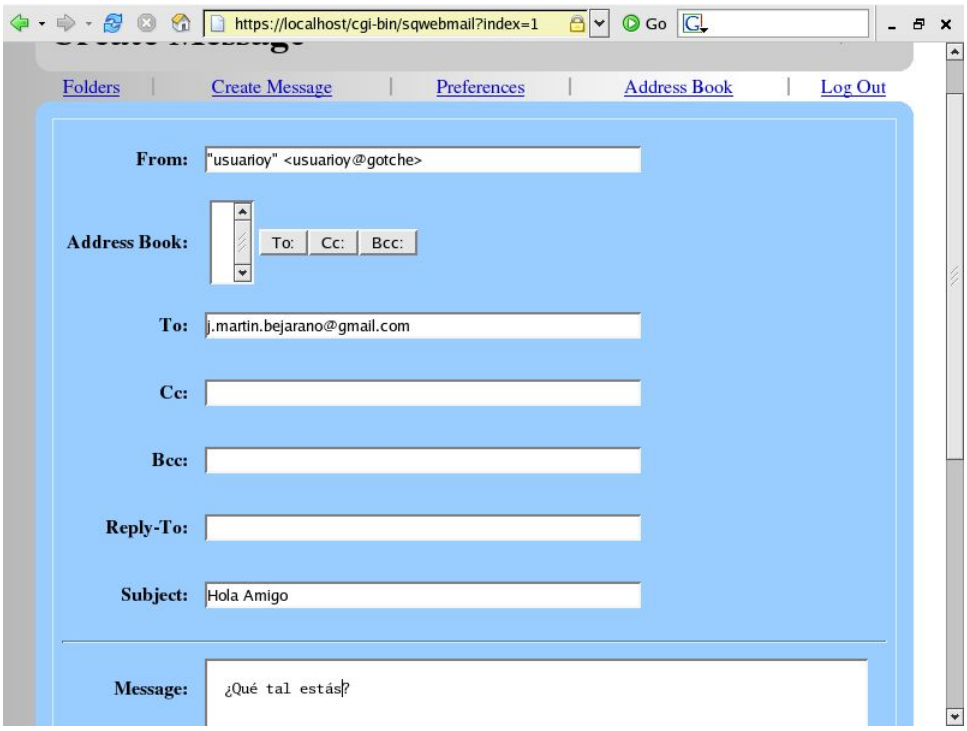

Figura 8.16 Usuarioy envía un correo externo

Resultado:

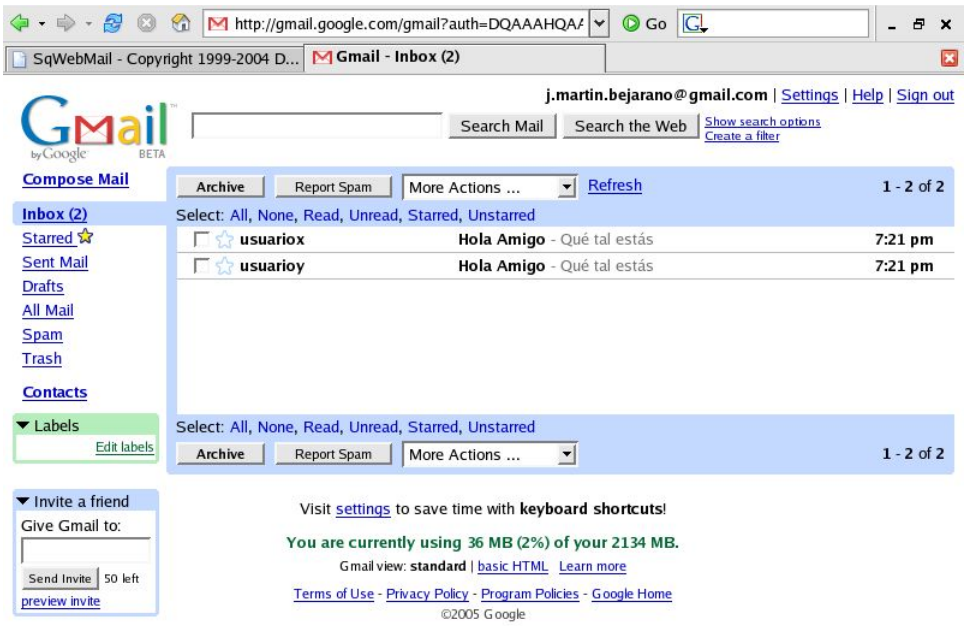

Figura 8.17 Recepción del mensaje por parte del usuario externo

5. Recibir correo en la cuenta de usuariox mandado por un usuario de gmail (correo externo)

   Resultado:

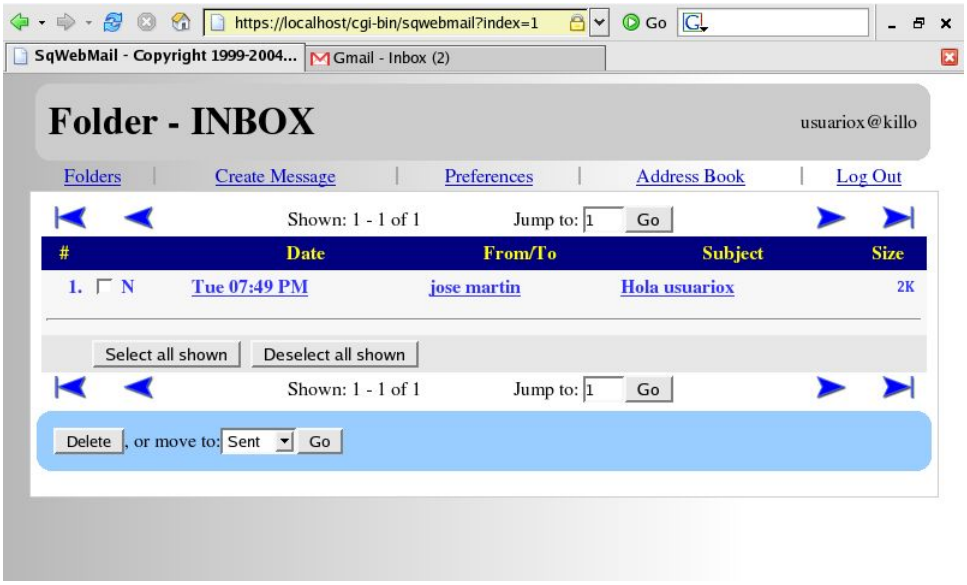

Figura 8.18 Usuariox recibe correo de un usuario externo (gmail)

6. Recibir correo en la cuenta de usuarioy mandado por un usuario de gmail (correo externo)

   Resultado:

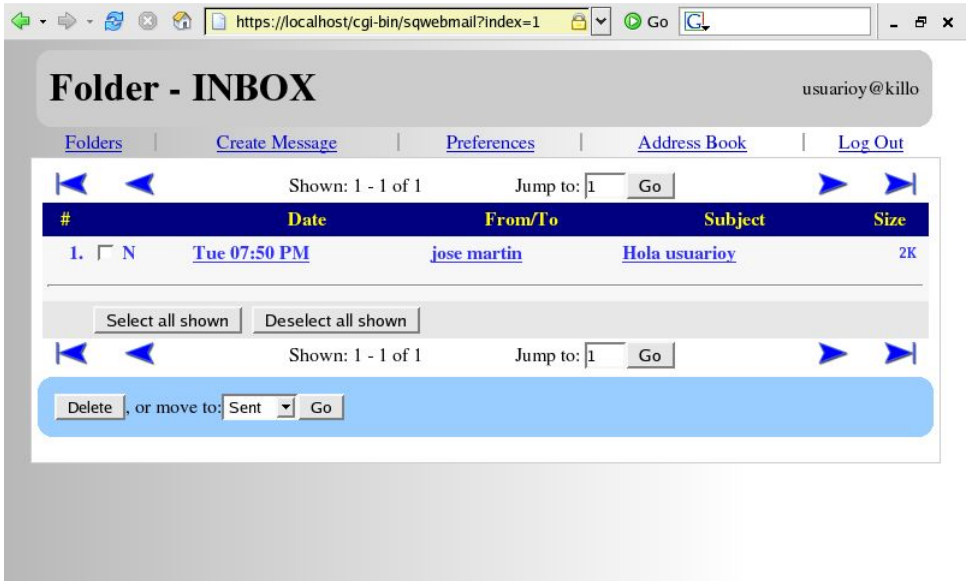

Figura 8.19 Usuarioy recibe correo de un usuario externo (gmail)

7. Mandar correo de usuariox a usuarioy (correo interno)

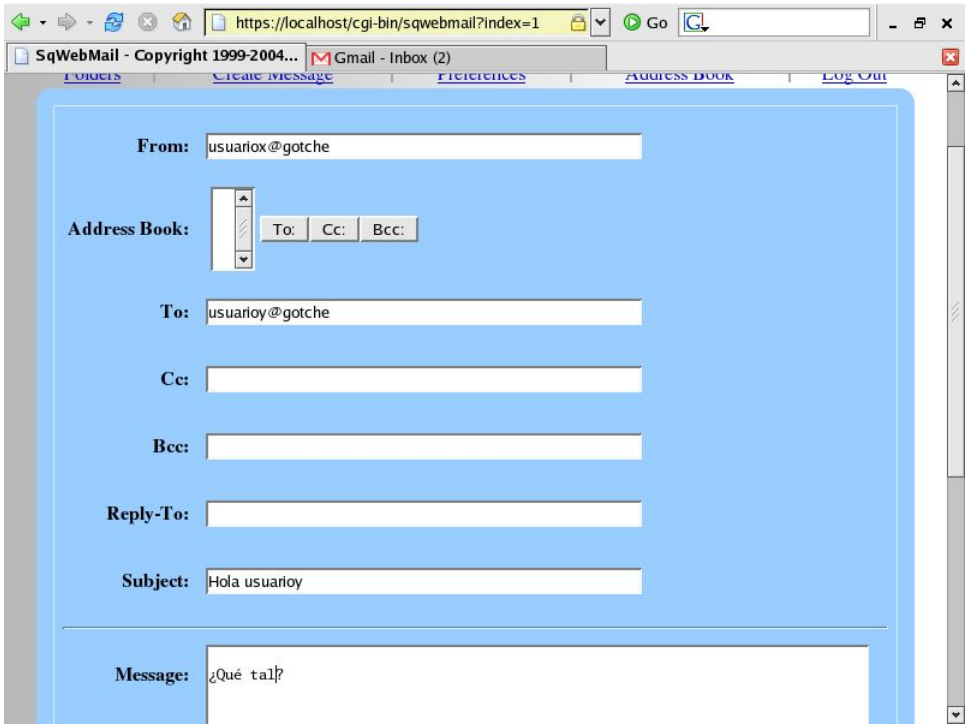

Figura 8.20 Usuariox envia correo a usuarioy

Resultado:

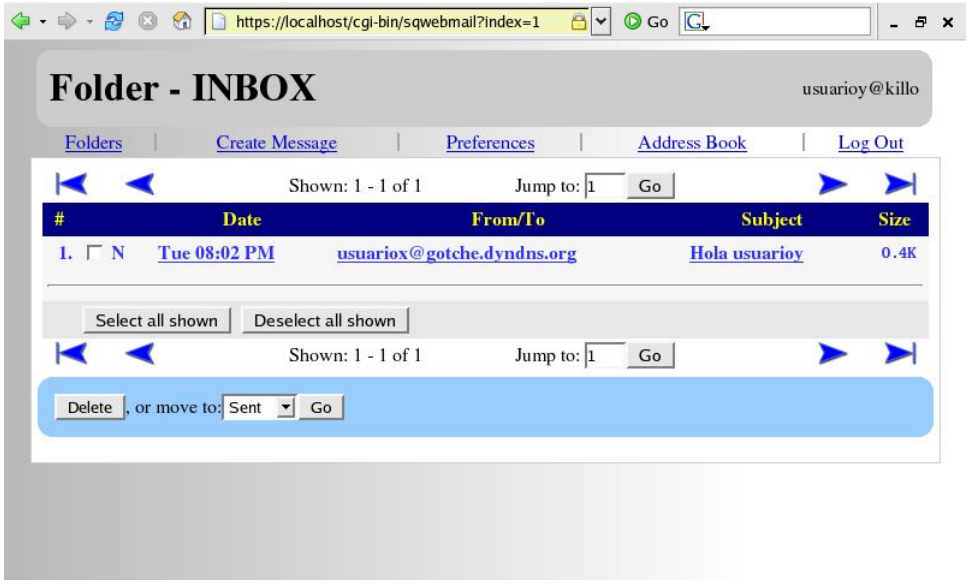

Figura 8.21 Usuarioy recibe correo de usuariox

8. Mandar correo de usuarioy a usuariox (correo interno)

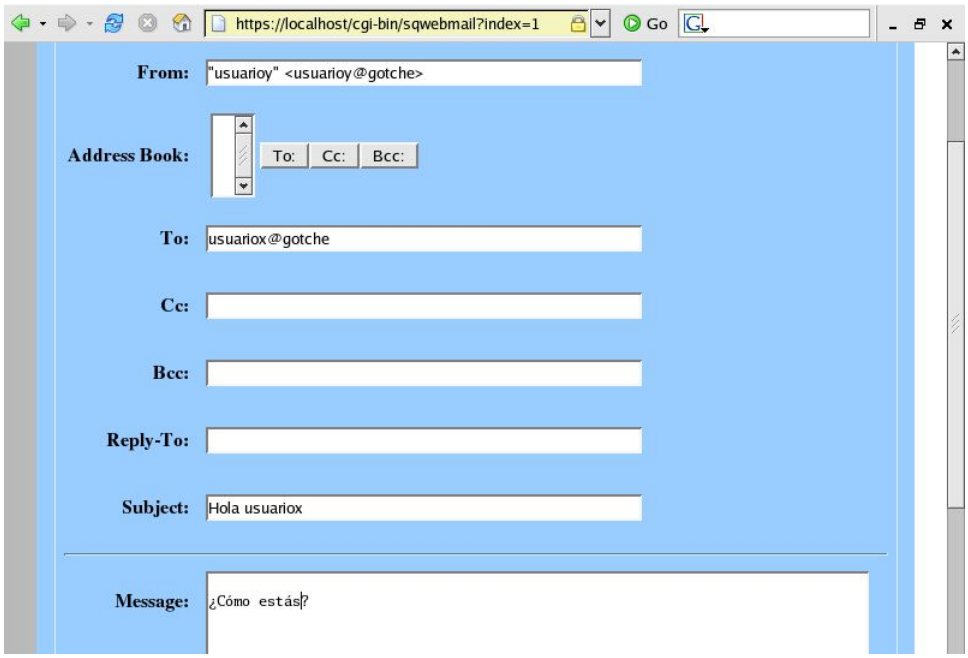

Figura 8.22 Usuarioy envia correo a usuariox

Resultado:

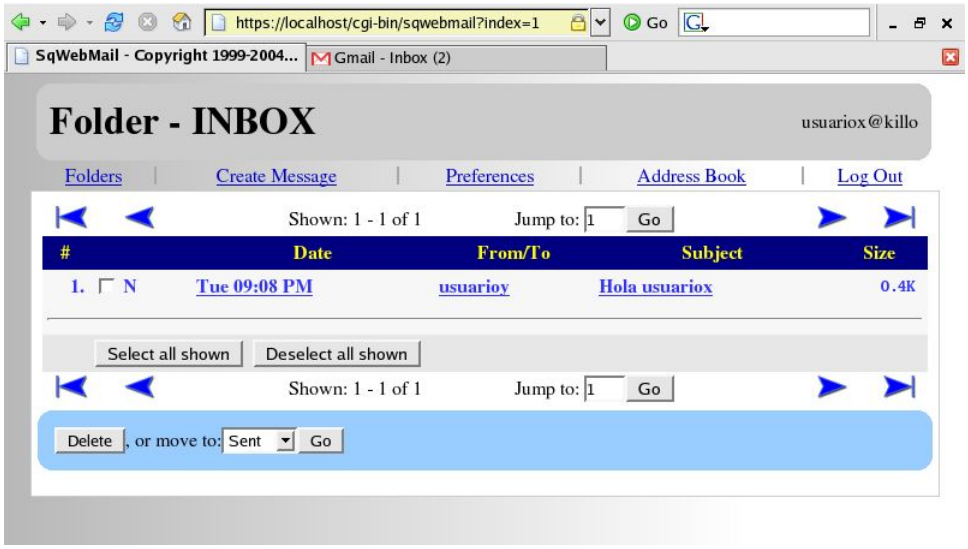

Figura 8.23 Usuariox recibe correo de usuarioy

Ejecución de las pruebas desde los distintos equipos

Nº de prueba	*Equipo gotche*	*Equipo fred*	*Equipo kinko*
1	✓	✓	✓
2	✓	✓	✓
3	✓	✓	✓
4	✓	✓	✓
5	✓	✓	✓
6	✓	✓	✓
7	✓	✓	✓
8	✓	✓	✓

8.9 Pruebas de seguridad

En cuanto a la seguridad nuestro principal referente va a estar la transmisión de las claves por la red. Esta transmisión no debe realizarse en texto plano, como ya hemos comentado en capítulos anteriores, sino cifrada.

Para ello se ha usado una capa segura (TLS o SSL). En lo que hemos llamado servicios Unix y en postgres se utiliza TLS para llevar a cabo lo dicho. Podemos comprobarlo en las capturas realizadas que se mostrarán a continuación.

Como ejemplo vamos a tomar el servicio "su" realizado desde el equipo fred. Para lograr la autenticación tanto el nombre de usuario como la contraseña deben cruzar la red. Si no tuviesemos la capa segura ocurriría lo que podemos ver en la siguiente captura.

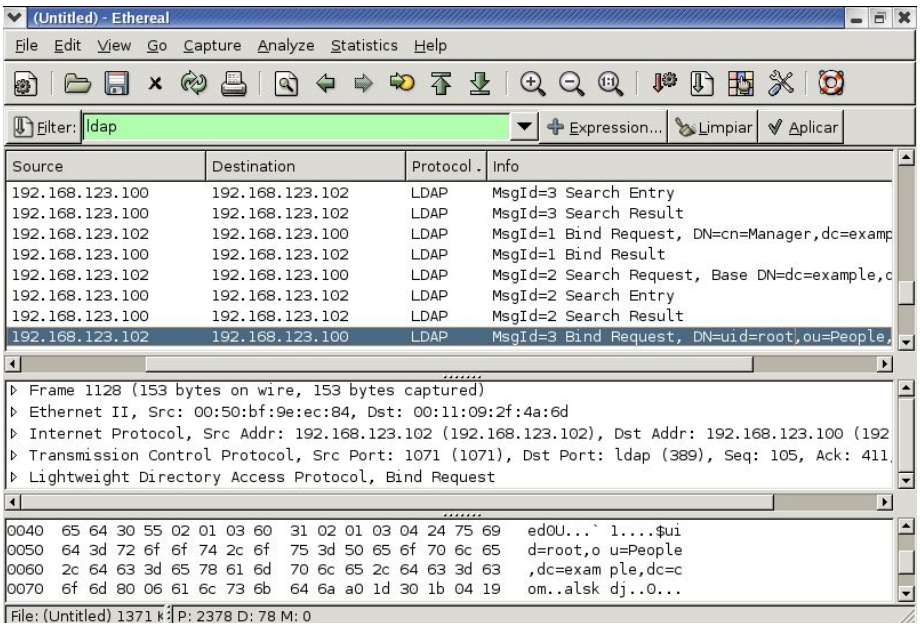

Figura 8.24 Captura LDAP no seguro

La contraseña, en este caso del usuario root, se vería comprometida. Se ha transmitido en texto plano.

Por el contrario, si usamos TLS, toda la información queda cifrada. Mostramos el anterior ejemplo, esta vez usando esta capa de seguridad.

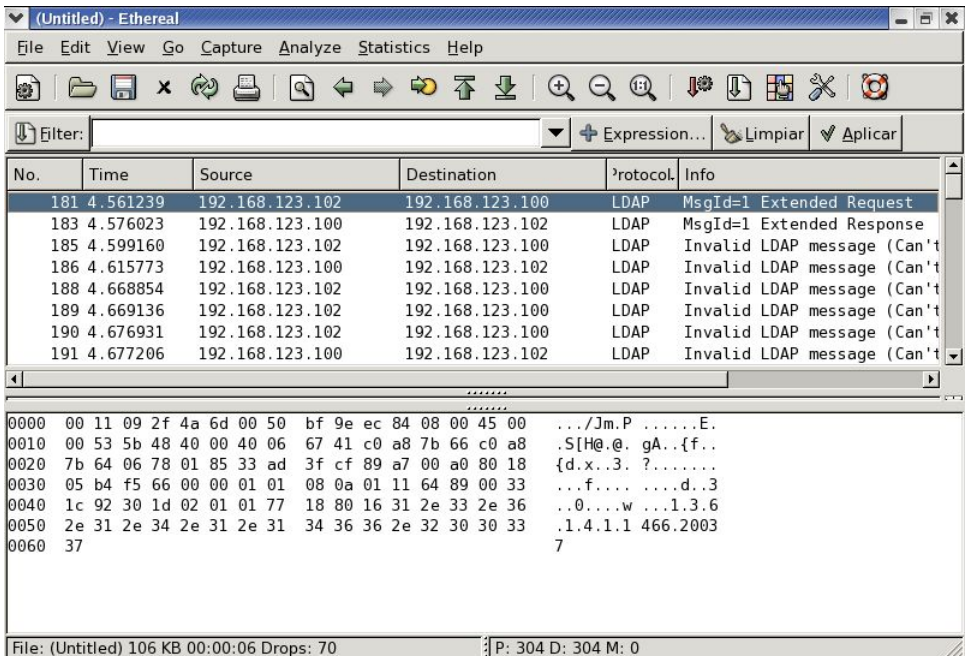

Figura 8.25 Captura LDAP seguro

En cuanto a qmail y SqWebMail nos encontramos en la misma situación, lo único que cambia es el protocolo. La transmisión de la clave se produciría en texto plano en http. Para ello tenemos la misma solución: usar SSL.

Podemos comprobar el funcionamiento de esta capa en las capturas de las figuras 7.6 y 7.8.