

REDES: Clase 18

CONFIGURACIÓN DE SAMBA

CONFIGURACIÓN DE SAMBA



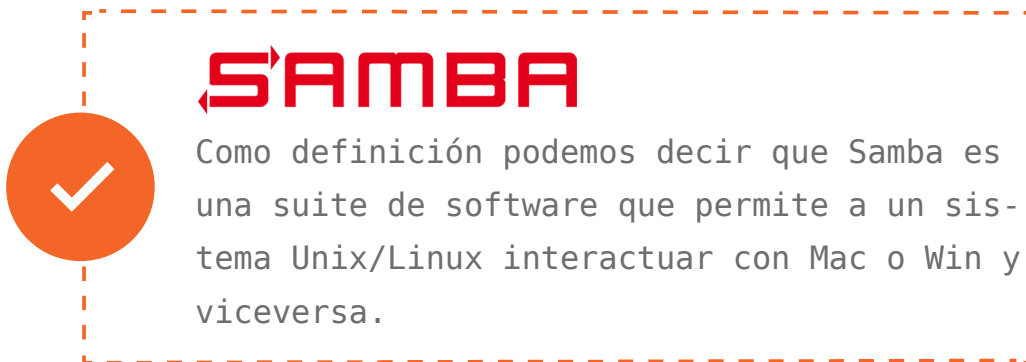
ÍNDICE

Introducción a SAMBA	3
¿Qué es Samba?	4
Los demonios de Samba	5
CIFS	6
NetBIOS	7
Obteniendo un nombre	9
Wins, NMBD y SMBD	11
Instalación de Samba	13
Archivo de configuración	15
Usuarios y passwords	16
Configuración de firewall	19
Testeando los demonios de Samba	20
El archivo de configuración de Samba	23
Tener en cuenta para la carga de variables	25
Variables Samba	26
La sección [global]	28
La sección [homes]	28
La sección [printers]	29
Configuración básica del servidor	29
Compartición de recursos de disco-usuarios	31
Compartición de recursos de disco-carpetas	34
Compartición avanzada	39
Permisos y atributos de GNU/Linux vs DOS	40

CONFIGURACIÓN DE SAMBA

Introducción a Samba

Mucho se habla de Samba, hasta existe el preconceito parcialmente cierto, casi erróneo, de que “Samba permite ver los archivos de Windows”. Antes de ahondar más en Samba, debemos decir que Samba soluciona problemas de interoperabilidad no solo en el sentido de Linux se adapta a Windows, sino que permite interconectar los 3 grandes (o los 3 que más encontraremos a nivel usuarios finales), Windows, Linux y Mac.



Existen varios componentes en Samba. Cada una de las piezas trabajan juntas para implementar tanto cliente como el servidor del protocolo CIFS. Este es un protocolo de red utilizado por Microsoft para la administración remota y acceso tanto a archivos como a impresoras. Pero noten que es un protocolo utilizado por Microsoft y es por este motivo que decimos que es erróneo pensar que Linux se adapta a Windows. Lo correcto es decir que ambos sistemas operativos utilizan el mismo protocolo.



CIFS no es un sistema de archivos ni tampoco es útil para internet. Sin embargo está difundido en redes Windows.

Existen razones por las cuales utilizaremos Samba en lugar de un servidor Windows; pero todas se basan no en pensar en Linux, sino en las bondades del software libre:

- Escalabilidad
- Flexibilidad
- Seguridad
- Sin costo

Esta última no está vinculado con software libre; pero es un beneficio. No hay que confundir libre con gratuito; pero lo que si van a tener es una comunidad libre dando soporte.

¿Qué es Samba?

Samba fue creado por Andrew Tridgell quien creó un file server que utilizaba un protocolo que se convertiría en SMB, el antecesor de CIFS. Unos años después, lo comenzó a distribuir como un producto libre. Pero para el nombre ejecutó este comando:

```
# grep -i '^s.*m.*b.*' /usr/dict/words
```

Y la respuesta fue:

```
salmonberry  
samba  
sawtimber  
scramble
```

Y así es como nació el nombre de Samba.



Como dijimos anteriormente, **Samba implementa el protocolo CIFS**, así que soportando este protocolo, Samba permite que máquinas corriendo sistemas operativos basados en Unix se puedan comunicar con sistemas operativos Windows y otros clientes CIFS activados.

Algunos servicios que ofrece Samba son:

- Compartir uno o más directorios
- Administración centralizada de impresoras.
- Autenticar clientes que se conectan a un dominio Windows.



La suite de Samba incluye herramientas que permiten a los usuarios de un sistema Unix acceder a carpetas de sistemas Windows y viceversa.

Los demonios de Samba

Básicamente Samba trabaja utilizando 3 demonios:

- 01 El primero es el demonio **smbd**. Este daemon administra la compartición de archivos e impresoras y realiza autenticación y autorización de los clientes.
- 02 El segundo es **nmbd** que manipula la registración del nombre NetBIOS de samba. Implementa una versión compatible del NetBIOS Name Server de Microsoft (WINS Server).
- 03 El tercero es el **winbindd**. Este daemon se comunica con los domain controllers para proveer información como grupos al cual pertenece el usuario. También provee una interfaz al WindowsLanManager para autenticación. También se lo conoce como NTLM.

CIFS



Los sistemas operativos de Microsoft trabajan con un protocolo para compartir recursos abierto conocido como CIFS. CIFS provee una API para manipular archivos y para implementar administración remota como cambio de contraseña.

Se puede creer que este es un protocolo que no tiene relación con su predecesor, el protocolo SMB; pero CIFS es la última variante de una larga línea de dialectos SMB. Muchos discuten que es la última versión de SMB. Es por este motivo que muchas veces encontrarán al protocolo SMB como sinónimos de CIFS, intercambiados o hasta a veces como una combinación (CIFS/SMB). Por simplicidad, vamos a referirnos como sinónimos SMB de CIFS.

Microsoft introdujo en Windows Vista una nueva variante de CIFS llamada SMB2; pero los detalles de este protocolo todavía están mostrándose. Como siempre, los desarrolladores de Samba continuaron trabajando para asegurar la compatibilidad con los sistemas operativos Windows.

CIFS es un protocolo orientado a conexión, stateful que se basa en tres servicios de red:

- Un servicio de nombre
- Medios para el envío de datagramas a un solo host o a un grupo de hosts
- Medios de establecer una conexión de largo término entre un cliente y un servidor.

Tanto Samba como Windows soportan el uso de servicios IP estándar para satisfacer lo que mencionamos anteriormente. ¿Qué queremos decir con esto?, por ejemplo el servicio de DNS, paquetes UDP, TCP que provee el soporte necesario para sesiones CIFS.



Aunque los clientes y servidores modernos CIFS pueden funcionar sin NetBIOS, **la mayoría provee una manera legacy de operación para comunicarse con viejas implementaciones CIFS.** Aquí vemos la relación entre el protocolo CIFS, hosts y servicios de red.

NetBIOS

La API de NetBIOS provee un diseño rudimentario para una aplicación para conectarse y compartir la información entre equipos. Es útil pensar en la API de NetBIOS como una extensión de red para la API de BIOS standard.




La **BIOS** contiene código de bajo nivel para desarrollar operaciones de filesystem en la computadora local. **NetBIOS** originalmente intercambiaba instrucciones con las máquinas IBM PC o redes Token Ring.

Igualmente requiere de un protocolo de transporte de bajo nivel para transportar sus requests de un equipo a otro o al próximo.

En finales de 1985, IBM lanzó un protocolo de este tipo que integró con la API de NetBIOS que se transformó en NetBEUI. NetBEUI fue diseñado para pequeñas LANs, y dejaba que cada equipo reclame un nombre de hasta 15 caracteres que no estuviera en uso. Con LANs pequeñas se pensaba en hasta 255 nodos... algo que en 1985 era difícil de alcanzar.

El protocolo NetBEUI fue muy popular con las aplicaciones de red. Más tarde, implementaciones de NetBIOS sobre el protocolo de transporte para redes Novell IPX también surgieron y se complementaron con NetBEUI. Sin embargo, y como es más que claro, la elección de stack de protocolos para la comunidad de internet fue TCP/IP, e implementar la API de NetBIOS sobre este protocolo se volvió una necesidad.

Como saben (me imagino...) TCP/IP utiliza números para representar las máquinas que son las direcciones IP y NetBIOS solo utiliza nombre. Esto fue un problema cuando se intentó integrar los dos protocolos. En 1987, la IETF publicó unos documentos (RFC 1001 y 1002) de estandarización que delineó como NetBIOS debía trabajar sobre una red TCP/IP. Esta serie de documentos aún maneja cada implementación existente hoy día, incluyendo aquellas proveídas por Microsoft con su sistema Windows, de la misma manera que la suite Samba.



El sistema de nombres NetBIOS resuelve el problema de nombre-a-dirección mencionado anteriormente, permitiendo que cada máquina declare un nombre específico en la red que puede ser traducido a una dirección IP.

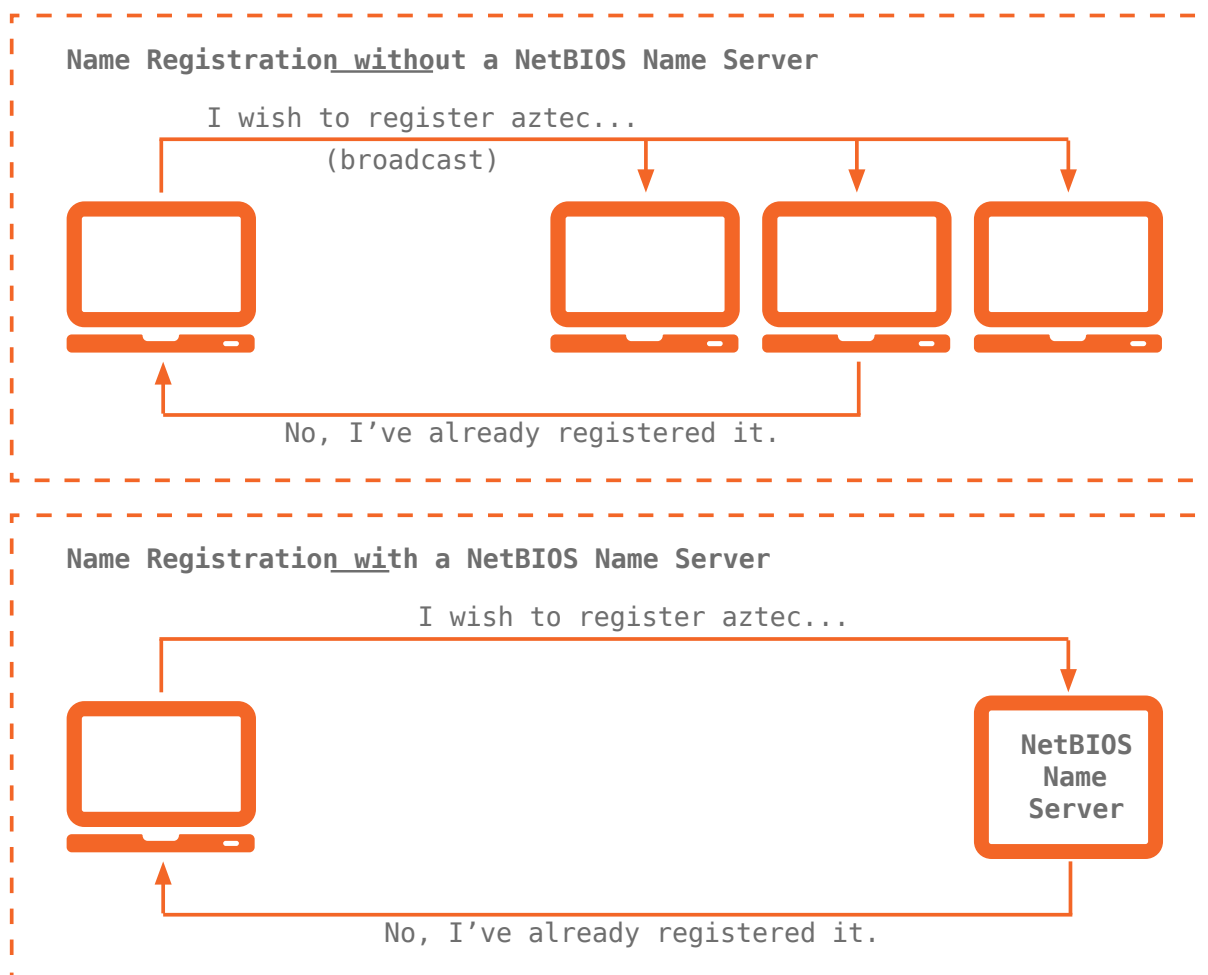
Con la omnipresencia de las redes TCP/IP y los DNS, que desempeña una función idéntica que los servicios NetBIOS es comprensible el por qué Microsoft decidió migrar de NetBIOS finalmente.

Obteniendo un nombre

En el mundo de NetBIOS, cuando cada computadora se pone online, la misma intenta obtener un nombre para ella misma; este proceso es llamado registración. Sin embargo, dos equipos no pueden reclamar el mismo nombre; este estado causará confusión para cualquier equipo que se quiera comunicar. Existen dos acercamientos diferentes para asegurarnos que esto no suceda:

- Permitir a cada computadora en la red defender su nombre en caso que otra computadora intente utilizarlo, ya que los nombres son asignado mediante broadcast.
- Utilizar un servidor WINS para mantener el track de que hosts existen en los diferentes segmentos de red que no son alcanzables a través de solicitudes de broadcast.

En la figura nos muestra una solicitud fallida de NetBIOS.



Como mencionamos anteriormente, debe existir una forma de resolver un nombre NetBIOS a una IP específica; este proceso es conocido como resolución de nombres. Para esto hay dos tipos de acercamiento:

- Hacer que cada equipo reporte su dirección IP cuando recibe un Broadcast solicitando por el nombre NetBIOS.
- Utilizar un servidor WINS para resolver el nombre NetBIOS a la dirección IP como veremos más adelante.



En el método de broadcast, cuando una máquina inicia, envía un mensaje declarando que desea registrar un nombre NetBIOS específico como propio. Si nadie objeta el uso de ese nombre, lo mantiene. De otro modo, si otra computadora de la subred local está utilizándolo, envía una respuesta de que ese nombre está siendo utilizado.



Este método es útil cuando un cliente inesperadamente se salió de la red ya que otro cliente puede reclamar el mismo nombre.

Wins, NMBD y SMBD

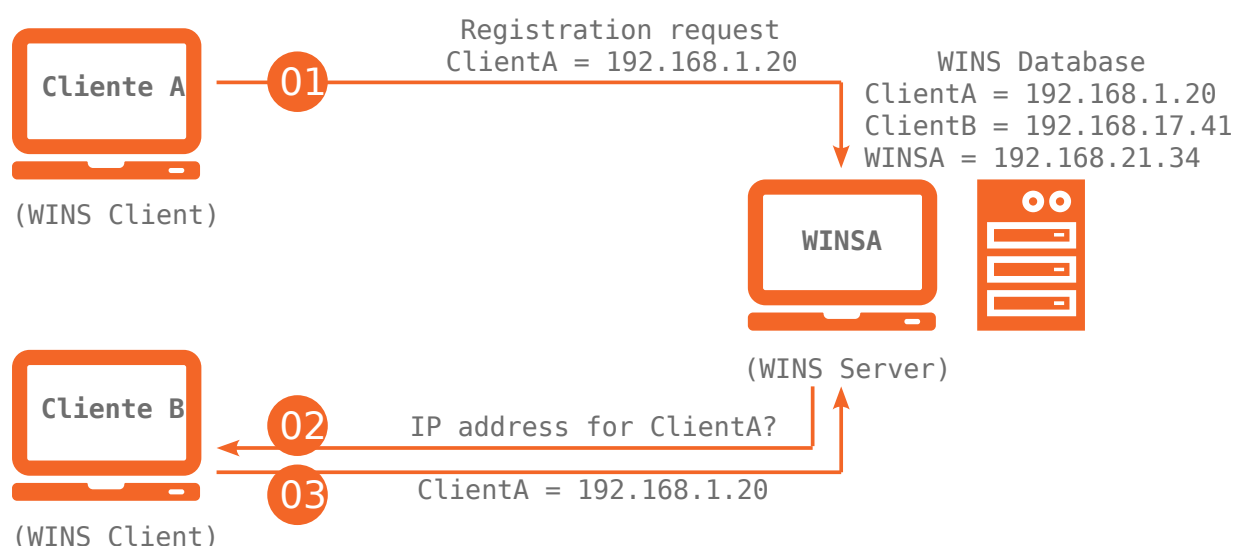
A continuación repasaremos estos tres ítems que es algo que vamos a ver mencionado bastante cuando hablamos de Samba.

Primero veamos WINS y el rol de un servidor WINS dentro de una red. La resolución de nombres WINS significa mapear de manera exitosa un nombre NetBIOS a una dirección IP.



Cuando un proceso NetBIOS se comunica con un servicio específico en una máquina específica, un único nombre es utilizado. Cuando un proceso NetBIOS se comunica con múltiples máquinas un nombre de grupo es utilizado.

Aunque NetBIOS puede ser utilizado en una red con otros protocolos de red que TCP/IP, WINS fue diseñado específicamente para soportar NetBIOS sobre TCP/IP (NetBT). WINS es requerido por cualquier entorno en donde los usuarios acceden a recursos que tienen nombres NetBIOS. Si no utilizamos WINS en la red (como va a ser nuestro caso), no se podrá conectar a un recurso remoto utilizando su nombre NetBIOS (OJO, nombre NetBIOS, no nombre de host), a menos que se utilice un archivo lmhosts donde dice dirección IP y nombre NetBIOS.



Aquí tenemos un gráfico que explica lo mencionado anteriormente. Tenemos tres equipos con nombres NetBIOS ClientA, ClientB y WINSA. Como observamos, dentro de la base de datos WINS se encuentran mapeados los nombres de NetBIOS a sus direcciones IP.

Como podemos ver, con WINS no ocurre ningún tipo de broadcast cuando la computadora desea registrar el nombre; el mensaje de registración es simplemente enviado directamente del cliente al servidor WINS. Este sistema es conocido como comunicación punto a punto, y es beneficiosa en redes con más de una subred. El mismo principio se aplica a la resolución DNS.



Sin WINS, la resolución de nombres NetBIOS también puede ser realizada como dijimos antes, hecho con mecanismo de broadcast. Utilizar WINS y comunicaciones punto a punto para este propósito es **menos costoso** en una red donde se puede inundar de solicitudes de resolución de nombre; pero esto es raro que suceda en las redes modernas.

Como mencionamos anteriormente, Samba contiene varios programas que sirven a propósitos diferentes; pero que se relacionan entre sí. Vamos a ver smb y nmb. Primero veamos nmbd.

El demonio **nmbd** es un servidor de nombres simples que reemplaza la funcionalidad de WINS. Este daemon escucha requests de nombres y provee la dirección IP correspondiente. Básicamente entiende y responde a los requests de NetBIOS sobre IP, como los producido por clientes SMB/CIFS. También participa en lo que es el armado del “Networking Neighborhood”.

Los clientes SMB/CIFS, cuando inician, pueden querer localizar un servidor SMB/CIFS. Si este es el caso, nmbd escuchará a esos requests, y si su propio nombre NetBIOS es especificado responderá con la dirección IP y número del host en el que está corriendo. Por defecto, el nombre NetBios es la parte de host del FQDN; pero esto puede ser sobrescrito en el archivo de configuración de Samba.



Nmbd puede ser utilizado también como servidor WINS o como WINS proxy que transmite el query de los clientes que no manejan el protocolo WINS a un servidor WINS.

El daemon **smbd** administra los recursos compartidos entre el servidor Samba y los clientes. Provee servicios de archivo e impresión a los cliente SMB/CIFS sobre una o más redes y maneja las notificaciones entre el servidor Samba y los clientes de red. Aparte también es responsable de la autenticación de usuarios, lockeo de recursos y la compartición de datos a través del protocolo SMB/CIFS.

Instalación de Samba

Como vimos anteriormente, Samba permite la interoperabilidad multiplataforma y esto lo hace una opción interesante para los administradores GNU/Linux, si se desea lograr una interoperabilidad simple y a nivel de LAN.

Para la instalación utilizaremos el paquete binario .deb de las distribuciones de Debian. Como ventaja podemos decir que no deberemos tratar con detalles de compilación y la actualización será más simple ya que está coordinada por el package manager de Debian.

Si deseamos saber si tenemos instalado previamente Samba podremos ejecutar el siguiente comando:

```
samba-server:~# dpkg -l | grep samba
```

Revisado esto, vamos a instalar Samba. Para esto utilizaré la máquina virtual subida al aula que dice “samba-server”. Esta máquina tiene un GNU/Linux sin Samba instalado que podremos utilizar; pero siéntanse libres de usar una distribución de Debian instalada por ustedes:

```
samba-server:~# apt-get install samba smbclient
```

Una vez hecho esto tendremos los demonios de Samba instalados. Para revisar el status de los mismos y que se encuentren corriendo correctamente podremos ejecutar el comando:

```
samba-server:~# /etc/init.d/samba status
[ ok ] nmbd is running.
[ ok ] smbd is running.
```

Donde mostrará el estado de cada uno de los demonios que hemos descrito anteriormente.

Ahora veremos las carpetas que ha creado Samba al momento de la instalación y la función que cumplen cada una de ellas:

- **/etc/samba**: fiel al estilo GNU/Linux, esta carpeta guarda los archivos de configuración.
- **/var/log/samba**: los logs creados por Samba.
- **/var/lib/samba**: el repositorio de la tdb o trivial database que se usa para mantener información de la configuración e información de estado como archivos abiertos.
- **/var/run**: aquí encontraremos los PID de los demonios smbd, nmbd y si lo activamos, winbind.

Ahora sí, podremos revisar que tenemos los paquetes de Samba instalados:

```
samba-server:/usr/share# dpkg -l | grep samba
ii  libcrypt-smbhash-perl      0.12-3
ii  samba                      2:3.6.6-6+deb7u4
ii  samba-common               2:3.6.6-6+deb7u4
ii  samba-common-bin           2:3.6.6-6+deb7u4
```

Y la versión de los mismos:

```
samba-server:/usr/share# smbd --version
```

ARCHIVO DE CONFIGURACIÓN

Como dijimos anteriormente, el archivo de configuración se encuentra dentro de la carpeta /etc/samba:

```
samba-server:~# cd /etc/samba/
```

Una vez dentro, y antes de abrir el archivo de configuración, haremos un backup del mismo ya que es muy simple confundirse al momento de ingresar parámetros.

```
samba-server:/etc/samba# cp smb.conf smb.conf_bkp  
samba-server:/etc/samba# vi smb.conf
```

La clave para configurar Samba es su archivo de configuración smb.conf. Este archivo puede ser muy simple o demasiado complejo. Por ahora vamos a copiar un archivo de ejemplo más simple que el que tenemos aquí y vamos a ir configurándolo de a poco. Por lo que vaciaremos el archivo de configuración:

```
samba-server:/etc/samba# > smb.conf
```

Y luego crearemos uno nuevo SOLO con el siguiente contenido:

```
samba-server:/etc/samba# vi smb.conf
```

```
[global]  
workgroup = INTRA-MYLinux  
  
#####RECURSOS COMPARTIDOS#####  
[test]  
comment = Test básico de compartición de recursos  
path = /data/test  
read only = no
```

Veamos un poco lo que fuimos haciendo. En primer punto tenemos la sección global que se encuentra separada por [] como todas las secciones. Esta sección contiene información que se aplicará a todo el servidor. Y luego tenemos la sección test que se aplica solo a los recursos compartidos. No se preocupen, abordaremos estos temas más adelante con mayor detalle.



Los parámetros en Samba son case insensitive por lo que el path puede ser escrito en mayúscula o minúscula. Tampoco importa los espacios en blanco por lo que ***read only*** es lo mismo que ***readonly***.

Nuestro servidor será parte del workgroup llamado INTRA-MYLINUX que está determinado por la variable workgroup.

Utilizaremos el recurso compartido llamado [test] para conectarnos con clientes windows. Por ahora, se puede completar el setup realizando los siguientes comandos como root en el servidor GNU/Linux:

```
samba-server:/etc/samba# mkdir -p /data/test  
samba-server:/etc/samba# chmod 1777 /data/test
```

Hemos puesto los permisos 1777 que es lo mismo que el tmp. Agregamos un permiso especial que nos indica que solo el owner del archivo puede eliminarlo. A excepción del root.



Aguarden, todavía no debemos reiniciar el servidor de Samba.

USUARIOS Y PASSWORDS



La encriptación de passwords es un área donde a veces suelen existir problemas de interoperabilidad entre Windows y GNU/Linux.

En este punto suele complicarse ya que utilizan un algoritmo no reversible. Esto significa que no se pueden reconvertir passwords Unix a los hashes LanMan o passwords NT o viceversa, por lo que Samba no tiene soporte para passwords que se encuentren en el /etc/passwd. O sea, Samba nunca se entera de nuestros usuarios de sistema. Inicialmente veremos un sistema básico de autenticación; pero más adelante veremos que lo podremos hacer mediante LDAP o mediante MySQL.

Por defecto, Samba encripta todos los passwords, por lo que solo para ser más claros agregaremos la siguiente línea al final de la sección global:

```
samba-server:~# vi /etc/samba/smb.conf
encrypt passwords = yes
```

Como explicamos anteriormente, deberemos crear usuarios que no estén relacionados con /etc/passwd y que cuyo password sea comprensible por los diferentes sistemas operativos, o sea que lo guarden estilo LMNT.



La **aplicación smbpasswd** es la que se utiliza para ingresar la combinación usuario/password de los usuarios Samba dentro de una base de datos encriptada.

Por ejemplo, para permitir a al usuario atos acceder a los recursos compartidos desde un cliente podremos utilizar el siguiente comando:

```
samba-server:~# adduser --no-create-home \
--disabled-password \
--disabled-login atos
```

```
samba-server:~# smbpasswd -a atos
New SMB password:
Retype new SMB password:
Added user atos.
```

Algo que se estarán preguntando es porque fue necesario crear un usuario en el sistema. Esto es porque la contraseña es la que no es compatible; pero Samba si revisa los usuarios para saber quién tiene que privilegios.



Dicho de otra manera, la autenticación es por parte de smbpasswd; pero no la autorización.

Si quisiéramos crear otro usuario, por ejemplo portos:

```
samba-server:~# adduser --no-create-home \  
--disabled-password \  
--disabled-login portos
```

```
samba-server:~# smbpasswd -a portos  
New SMB password:  
Retype new SMB password:  
Added user portos.
```

Más adelante veremos cómo eliminar o desactivar usuarios. Si queremos ver los usuarios que hemos creado:

```
samba-server:~# pdbedit -L  
atos:1001:  
portos:1002:
```



Una vez que realizamos todas las configuraciones, debemos de **testear nuestro archivo de configuración**. Para esto vamos a utilizar un parseador denominado **testparm** que lo que hace es examinar el archivo **smb.conf** buscando errores de sintaxis y reportes.

Por ejemplo podremos ejecutarlo:

```
samba-server:~# testparm -s /etc/samba/smb.conf | more
Processing section "[test]"
Loaded services file OK.
Server role: ROLE_STANDALONE
[global]
    workgroup = INTRA-MY LINUX
    idmap config * : backend = tdb

[test]
    comment = Test básico de compartición de recursos
    path = /data/test
    read only = No
```

CONFIGURACIÓN DE FIREWALL

Como cualquier servicio que corre bajo TCP/IP, el servicio de red samba puede ser accedido a través de internet. Aquí tenemos los elementos a habilitar en el firewall para dejar pasar este tráfico:

- **137/udp**: es utilizado por el browser de red netbios nmbd.
- **138/udp**: es utilizado por el servicio de nombre de NetBios
- **139/tcp**: es utilizado para compartir impresoras y archivos y otras operaciones
- **445/tcp**: es utilizado por clientes a partir de Windows 2000.... si, en algunos lugares aún se los puede encontrar.
- **901/tcp**: el puerto de swat que veremos más adelante.



Como dijimos anteriormente, **SMB/CIFS no está preparado para internet, pero si para una LAN**. Por lo que lo recomendado es que si queremos acceder desde fuera de nuestra LAN utilicemos algún tipo de VPN.

Luego, debemos verificar que están en escuchas estos puertos TCP y que los UDP están levantados:

```
samba-server:~# /etc/init.d/samba start
samba-server:~# netstat -putan | grep mbd
```

TESTEANDO LOS DEMONIOS DE SAMBA

Ya casi finalizamos el setup del servidor Samba. Todo lo que nos queda hacer es asegurarnos que todo está funcionando correctamente en un principio como debería. Una manera conveniente de realizar esto es utilizar la aplicación smbclient para examinar que es lo que nos está ofreciendo; u ofreciendo a la red, el equipo.



Es muy importante que el host samba-server figure dentro del archivo /etc/hosts.

```
samba-server:~# smbclient -L localhost -N
```

```
Anonymous login successful
```

```
Domain=[INTRA-MY LINUX] OS=[Unix] Server=[Samba 4.1.17]
```

Sharename	Type	Comment
-----	----	-----
test	Disk	Test básico de compartición de
IPC\$	IPC	IPC Service (Samba 4.1.17)

```
Anonymous login successful
```

```
Domain=[INTRA-MY LINUX] OS=[Unix] Server=[Samba 4.1.17]
```

Server	Comment
-----	-----
SAMBA-SERVER	Samba 4.1.17

Workgroup	Master
-----	-----
INTRA-MY LINUX	SAMBA-SERVER

Vayamos por parte:

- **-L**: lista lo que ofrece a la red el host localhost
- **-N**: saltea el password. O sea, que no pregunte password.

Y vemos que aparece el recurso compartido test. También vemos que aparece algo llamado IPC, esto es lo que nos permite listar los recursos de manera anónima y esto puede ser restringido. Debajo tenemos los datos del servidor propiamente dicho.

Si quisiéramos listar los recursos; pero necesitamos de usuario y contraseña podemos hacer lo siguiente:

```
samba-server:~# smbclient -L localhost -U atos
```

```
Enter atos's password:
```



El modificado **-U** nos exige el ingreso de un usuario seguido de un password. Pero que pasa con Windows. Bien con Windows también podremos ingresar.

Ahora debemos probar el ingreso a lo que son los recursos. Para esto probaremos primero desde Linux:

```
samba-server:~# smbclient //localhost/test -U atos
```

```
Enter atos's password:
```

```
Domain=[INTRA-MYLINUX] OS=[Unix] Server=[Samba 4.1.17]
```

```
smb: \>
```

Lo primero que tienen que notar es que debimos poner el hostname seguido del recurso compartido. Esto lo podemos tomar del listado anterior, en la columna Sharename. Luego, veremos que se nos habilita un prompt `smb:\>`. Esto nos indicará que Samba nos ha dado un prompt para poder trabajar.

A continuación les adjunto una planilla con los comandos para trabajar con el shell de Samba:

- **pwd**: directorio donde estamos parados dentro del servidor
- **lcd**: cambios de directorio en el equipo local.
- **put**: este comando nos permite subir archivos al servidor (debemos colocar el nombre completo del archivo).
- **mput**: este comando nos permite subir varios archivos al servidor. Podemos usar caracteres comodín, como por ejemplo el caracter “*”.
- **get**: este comando nos va a permitir descargar archivos desde el servidor teniendo en cuenta que debemos colocar el nombre completo.
- **mget**: este comando nos va a permitir subir archivos al servidor, también como en el caso del comando put, nos permite usar caracteres comodín, como el “*”.
- **del**: Nos permite eliminar archivos.
- **ls**: lista los archivos remotos.

EL ARCHIVO DE CONFIGURACIÓN DE SAMBA



El archivo de configuración de Samba smb.conf, utiliza el mismo formato que los archivos de configuración típicos de Debian con el formato `variable=valor`.

Volvamos a modificar el archivo de configuración de Samba de la siguiente manera (encontrarán el archivo dentro del aula como smb.conf_2):

```
[global]
## core networking options
netbios name = samba-server
workgroup = INTRA-MY LINUX
encrypt passwords = yes

## netbios name service settings
wins support = no
## logging
log level = 1
max log size = 1000

## default service options
read only = no
[homes]
browseable = no

[test]
comment = Solo testing
path = /data/test
read only = no
```

Y reiniciamos:

```
samba-server:~# /etc/init.d/samba restart
```

La estructura del archivo de configuración, visto desde un punto de vista macro, podemos dividirlo en:

```
[global]
...
[homes]
...
[test]
...
```



Los nombres dentro de los [] delimitan las secciones dentro del archivo de configuración de Samba.



Las secciones corresponden al nombre de cada recurso compartido o servicios que pueden ser vistos por **[test]** y **[homes]**.

Por ejemplo, las secciones test y homes son recursos de red compartidos, contienen las opciones que mapean directorios específicos del servidor Samba. Todas las secciones definidas en el archivo smb.conf, con la excepción de la sección [global], son recursos disponibles como impresoras o archivos.

Estas secciones ayudan a agrupar los elementos dentro de un mismo scope. Existen dos tipos de scopes: Global, que se definen como comportamiento estándar para todos los servicios, a menos que se sobrescriba dentro de un share específico (por ejemplo test). Como los parámetros se parsean de modo top-down, si se setea la misma opción más de una vez en la misma sección, el último valor especificado es el único que se aplicará y estarán dentro de una de las siguientes categorías:

- **Boolean:** Yes/No, True/Falso, 1/0
- **Integer:** entero depende del parámetro.
- **Character string or list:** son strings del tipo free-form.
- **Enumerated:** se acepta un valor de una lista posible.
- **Plugins:** por ejemplo el uso dentro de un servidor de LDAP o MySQL como veremos durante el curso.

TENER EN CUENTA PARA LA CARGA DE VARIABLES

Algo a tener en cuenta al momento de cargar las variable son los siguientes ítems. Por un lado, los parámetros son no sensitivos ni a mayúsculas ni a espacio.

Los delimitadores por defecto en el archivo smb.conf son:

- **Coma:** atos, portos, aramis.
- **Continuación de línea:** si tenemos una línea que ocupa más de una línea podremos escapearlo de la siguiente manera:

```
"Mary tenía un corderito \  
que lindo corderito"
```

Dada la contrabarra (o backslash)
- **Comentarios:** se pueden insertar comentarios de a una línea por vez iniciando con el # o el punto y coma (;).

VARIABLES SAMBA

Vamos a ver el uso de otro tipo de variables dentro de Samba .

VARIABLE	DEFINITIONS
----------	-------------

Client Variables

%a	Client's architecture (see Table 4-1).
%i	IP address of the interface on the server to which the client connected.
%I	Client's IP address (e. g. 172.16.1.2).
%m	Client's NetBIOS name.
%M	Client's DNS name (defaults to the value of %I if hostname lookups = no).

User Variables

%u	Current Unix username (requires a connection to a share).
%U	Username transmitted by the client in the initial authentication request.
%D	User's domain (e. g., the string DOM-A in DOM-A\user).
%H	Home directory of %u.
&g	Primary group %u.
%G	Primary group %U.

Share Variables	Share Variables
-----------------	-----------------

%S	Current share's name.
%P	Current share's root directory.
%p	Automounter's path to the share's root directory, if different from %P.

Server Variables

- `%d` — Current server process ID.
- `%h` — Samba server's DNS hostname.
- `%L` — Samba server's NetBIOS name sent by the client in the NetBIOS session request.
- `%N` — Home directory server, from the automount map.
- `%v` — Samba version.

Miscellaneous Variables

- `%R` — The SMB protocol level that was negotiated.
- `%T` — The current date and time.
- `%(var)` — the value of environment variable var.

Tomado de Using Samba 3rd Edition O'Reilly.

Supongamos que se quiere compartir el home de los usuarios dentro de un directorio Unix; pero por un tema de prolijidad (y salud mental para el sysadmin) no queremos crear una entrada por cada uno de los usuarios.

Para esto utilizaremos la variable `%U`, por lo que podríamos por ejemplo colocar la siguiente línea:

```
[homes]
path = /data/home/%U
```

Donde previamente creamos la carpeta para el usuario atos, seteamos los permisos, configuramos su home. No se preocupen, veremos esto más adelante como ejemplo, esto fue simplemente para mostrar un ejemplo de uso de variables.

LA SECCIÓN [GLOBAL]

Esta sección fue descrita anteriormente y tiene dos propósitos fundamentales que es setear parámetros globales del servidor.



Cuando Samba lee su archivo de configuración, crea un servicio especial interno que contiene los parámetros por defecto con sus valores. Cualquier parámetro definido en la sección global es agregado a esta lista.

Por ejemplo, cuando un cliente se conecta al share [test], que contiene la línea “read only = no”, el share será marcado como readonly no sin importar lo que diga la variable global.

LA SECCIÓN [HOMES]



Si un cliente intenta conectarse a un share que no figura en el **smb.conf**, Samba busca por defecto un share llamado **[homes]** en el archivo de configuración.

Si existe el share [homes], smbd comienza a buscar para validar que el share name sea un usuario de GNU/Linux. Si ese usuario aparece en la base de autenticación (/etc/passwd, LDAP) en el servidor, entonces Samba crea un share temporario que contiene el username con todos los atributos del share [home].

LA SECCIÓN [PRINTERS]



Esta sección es para compartir los recursos de impresora.

No ahondaremos en esta sección durante el curso ya que la compartición de impresora está quedando obsoleta debido a la proliferación de impresoras de red con la capacidad de que el mismo equipo comparta un servicio de impresión, en lugar de eso preferimos focalizar en los recursos compartidos de archivos.

CONFIGURACIÓN BÁSICA DEL SERVIDOR

Hasta ahora creamos el archivo de configuración; pero no detallamos las líneas. Vamos a meternos un poco más en este aspecto:

```
samba-server:~# cat /etc/samba/smb.conf
```

Dentro de la sección global:

```
[global]
```

Seteamos el nombre de NetBios del servidor:

```
## core networking options
netbios name = samba-server
```

Su workgroup:

```
workgroup = INTRA-MYLINUX
```

Y que guarde las contraseñas de manera encriptada:

```
encrypt passwords = yes
```

Le quitamos el soporte para Wins:

```
## netbios name service settings
wins support = no
```

Y seteamos el nivel de logging, entre 0 y 10. Más de 3 no es recomendable junto con el tamaño máximo de cada archivo de logs en KB. Superado este tamaño, se renombra como bak:

```
## logging
log level = 1
max log size = 1000
```

Por defecto los servicios compartidos serán solo read only:

```
## default service options
read only = no
```

El siguiente parámetro es altamente recomendable que este dentro de la sección [homes] ya que impide que el recurso sea listado dentro de los recursos que comparte el server (recordar el smbclient -L):

```
[homes]
browseable = no
```

Por último, tenemos el recurso que comparte. El comment se refiere a la descripción del recurso en cuestión:

```
[test]
comment = Solo testing
path = /data/test
read only = no
```

Vamos a agregar algunos ítems:

```
samba-server:~# vi /etc/samba/smb.conf
```

Primero podemos agregar la descripción del servidor que aparecerá al momento de hacer el smbclient -L. Dentro de la sección global, podemos ponerlo debajo de workgroup:

```
server string = INTRA-MYLinux file server(Samba %v)
```

Inmediatamente debajo agregamos la siguiente línea. Lo hemos visto anteriormente y se refiere a la autenticación que realiza samba de usuario y contraseña:

```
security = user
```

Y reiniciamos el servicio:

```
samba-server:~# /etc/init.d/samba restart
```

COMPARTICIÓN DE RECURSOS DE DISCO - USUARIOS

A este punto ya tenemos una teoría sólida que nos ayuda a entender el funcionamiento de Samba; por lo que en este momento empezaremos a meternos de lleno en la compartición de recursos.

El primer ejemplo que veremos será la compartición de recursos de home. Si recuerdan, la sección [homes] nos permitía poner recursos que se asociaban con el nombre de usuario. Volvamos a revisar que usuarios tenemos hasta ahora:

```
samba-server:~# pdbedit -L
```

Como podemos ver tenemos dos usuarios creados. Ahora volveremos a ingresar al archivo de configuración:

```
samba-server:/etc/samba# vi /etc/samba/smb.conf
```

E iremos a la sección de [homes] y colocaremos lo siguiente:

```
[homes]
browseable = no
comment = Home de usuario %u
writable = yes
```

Donde:

- **writable**: son los permisos de escritura activados para la carpeta home.
- **comment**: es el comentario del recurso. Se coloca una descripción.

Si vamos a manejar los homes de los usuarios mediante Samba es recomendable realizar dos acciones:

- 01 Primero, vamos a crear unas carpetas especiales donde se almacenarán los homes de los usuarios; pero de Samba. De esta manera estamos separando lo que son los usuarios del sistema de los usuarios de Samba:

```
samba-server:/etc/samba# mkdir -p /data/home/atos
samba-server:/etc/samba# mkdir -p /data/home/portos
```

Ahora asignemos los permisos y owner correspondientes a las carpetas:

```
samba-server:/etc/samba# chown atos:atos /data/home/atos
samba-server:/etc/samba#chown portos:portos /data/home/portos
```

- 02 Ahora viene la segunda parte, que es indicarle a Samba que utilice como home de los usuarios la carpeta específica de un usuario. Pero ¿cómo hacemos esto?, utilizando las variables.

Ingresaremos nuevamente al archivo de configuración de Samba, y en la sección [home] agregamos la siguiente línea:

```
path = /data/home/%U
```

Y reiniciamos Samba:

```
samba-server:/etc/samba# /etc/init.d/samba restart
```


Ahora si, primero revisemos con Linux que nos lista con el usuario portos:

```
samba-server:/etc/samba# smbclient -L 172.16.0.138 -U portos
```

...

```
portos          Disk          Home de usuario portos
```

Y comparémoslo con lo que nos lista con el usuario atos utilizando una máquina Windows:



Si quisiéramos agregar un nuevo usuario, simplemente serían dos comandos. El primero que lo agrega como usuario del sistema:

```
samba-server:/etc/samba# adduser --home /data/home/aramis \
--disabled-login \
aramis
```

Y luego lo agregaremos como usuario de samba:

```
samba-server:/etc/samba# smbpasswd -a aramis
samba-server:/etc/samba# mkdir -p /data/home/aramis
samba-server:/etc/samba# chown portos:portos /data/home/portos
```

COMPARTICIÓN DE RECURSOS DE DISCO - CARPETAS

Por un lado, ya tendremos configurado lo que son las carpetas que compartirá por defecto a los diferentes usuarios de Samba; pero también, podemos llegar a querer tener carpetas compartidas para diferentes recursos.

Por ejemplo, supongamos que queremos tener una carpeta por cada sector de la empresa, que después se transformará en los famosos discos compartidos.



Esto estará asociado a diferentes usuarios y a veces tendremos la necesidad de que algunas sean de solo lectura. O que algunos usuarios tengan solo lectura.

Esto lo iremos viendo durante esta parte de la clase, configurando lo necesario y explicando por qué se utiliza cada cosa.

Lo primero que haremos será crear las diferentes carpetas. Sigamos con el ejemplo de una carpeta compartida por cada sector, por lo que la nomenclatura al momento de la creación de la carpeta que voy a utilizar es la siguiente:

```
sector-DISCO
```

Entonces crearemos nuestro primer disco. Por ejemplo, administración que llevará la letra Z. Como verán, lo primero que se hará es crear la carpeta que compartiré siendo el disco:

```
root@samba-server:~# mkdir -p /data/discos/administracion-Z
```

Luego ingresaremos a nuestro archivo de configuración y agregaremos lo siguiente al final:

```
[administracion-Z]
comment = Disco compartido del sector de administracion
path = /data/discos/administracion-Z
read only = no
```

Por ahora no cambio nada. Guardemos, y reiniciemos el servicio e intentemos subir algún archivo:

```
root@samba-server:~# /etc/init.d/samba restart
root@samba-server:~# echo "HOLA MUNDO" > upload.samba
root@samba-server:~# smbclient //172.16.0.138/administracion-Z -U aramis
smb: \> put upload.samba
NT_STATUS_ACCESS_DENIED opening remote file \upload.samba
```

Acceso denegado. Algo muy importante al momento de tener en cuenta cuando trabajamos con Samba es que al autenticarnos con usuario aramis, estamos autorizados como usuario aramis.

Si nos fijamos, la carpeta administracion-Z tiene otros permisos:

```
root@samba-server:~# ls -l /data/discos/
drwxr-xr-x 2 root root 4096 Feb  4 10:45 administracion-Z
```



No tenemos permisos de escritura. Esto es algo muy importante a tener en cuenta en el momento de trabajar con Samba. JAMÁS los privilegios que puede dar un servicio superan al sistema operativo. O sea, los permisos de Samba nunca superarán a los permisos de GNU/Linux.

Para solucionar esto se puede utilizar cualquier lógica de negocio y darle los permisos correspondientes. La lógica de negocio que aplico yo es crear un grupo con permisos de lectura-escritura y otro grupo de solo lectura con el mismo nombre del disco, y a ese grupo ir agregando los usuarios dependiendo de los permisos que quiera que tengan. De esta manera, si un usuario quiero agregarlo a más de un disco, simplemente lo agrego en el grupo, por lo que primero crearemos los grupos:

```
root@samba-server:~# groupadd administracion-Z_RW
root@samba-server:~# groupadd administracion-Z_R0
```

Luego agreguemos el usuario al grupo:

```
root@samba-server:~# usermod -aG administracion-Z_RW aramis
```

Paso siguiente, deberemos modificar los grupos y permisos de la carpeta administracion-Z:

```
samba-server:~# chown \  
root:administracion-Z_RW /data/discos/administracion-Z
```

Y le damos permisos de escritura al grupo:

```
root@samba-server:/data/discos# chmod 775 administracion-Z
```

Y ahora debemos de modificar la sección dentro del archivo de configuración de samba para que quede de la siguiente manera:

```
[administracion-Z]  
comment = Disco compartido del sector de administracion  
path = /data/discos/administracion-Z  
read only = no  
valid users = @administracion-Z_RW,@administracion-Z_R0  
write list = @administracion-Z_RW
```

Como pueden ver, agregamos 2 líneas más:

- **valid users:** Hacemos que este share esté disponibles para ciertos usuarios. Anteponer el símbolo @ implica que no estamos hablando de usuarios sino de grupos. Podemos enumerar los grupos utilizando la “,”.
- **write list:** los usuarios aquí especificados tienen permiso de escritura dentro del share.

Ahora si, llegó el momento de probar el share:

```
root@samba-server:~# /etc/init.d/samba restart
```

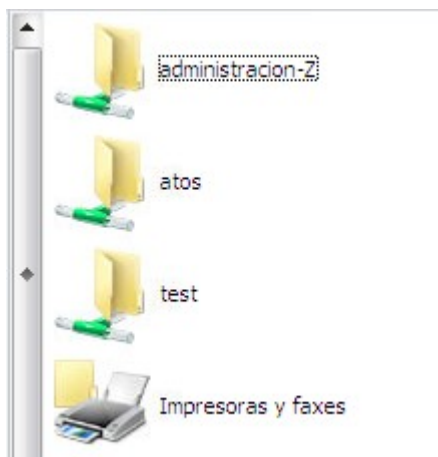
Y subamos el archivo:

```
root@samba-server:~# smbclient //172.16.0.138/administracion-Z -U aramis
smb: \> put upload.samba
putting file upload.samba as \upload.samba (0.1 kb/s) (average 0.1 kb/s)
```

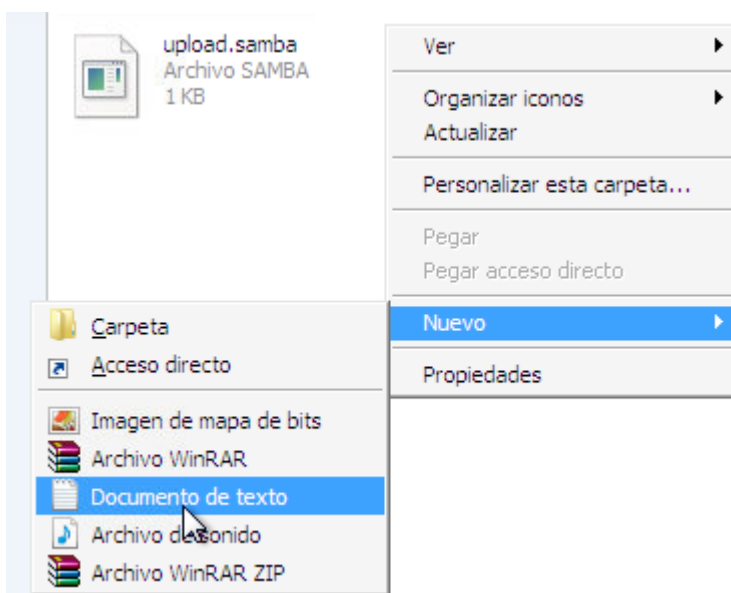
Y pudimos subir el archivo sin problemas. Veamos qué pasa con un usuario que no está en el grupo administracion-Z_RW; pero si en administracion-Z_RO:

```
root@samba-server:~# usermod -aG administracion-Z_RO atos
```

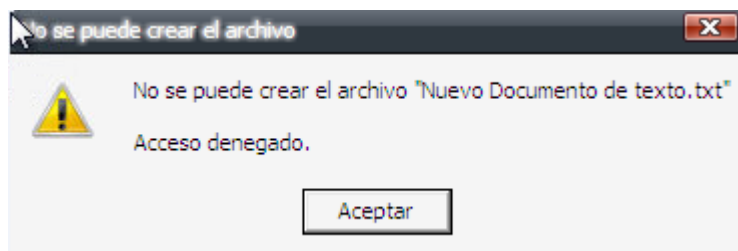
Y ahora intentemos acceder mediante Windows con el usuario. Ingreseemos al servidor como hicimos anteriormente:



Y hagamos doble click en administracion-Z. Podremos acceder sin inconvenientes; pero al momento de querer escribir algo:



Nos aparecerá el siguiente error:



Por lo que hasta ahora aprendimos a crear recursos compartidos los cuales tendremos la libertad de darles permisos de escritura o lectura para seguridad, controlando quienes tienen esos privilegios a través de la asignación de los mismos a los grupos correspondientes.

Todavía no hemos terminado para dejarlo funcional como querríamos. Esto es porque el recurso subido por aramis tiene los siguientes permisos en el servidor:

```
root@samba-server:~# ls -l /data/discos/administracion-Z/  
-rwxr--r-- 1 aramis aramis 11 Feb  4 11:59 upload.samba
```



Por lo tanto si, por ejemplo, agrego al **usuario portos** al **grupo administracion-Z_RW** podremos escribir en el **recurso compartido administracion-Z**; pero no podremos editar los archivos subidos por otro usuario ya que el owner y los permisos que asigna por defecto no lo permiten.

COMPARTICIÓN AVANZADA

Planteado el problema anterior vamos a ver algunos conceptos de compartición avanzada de recursos. Como solución vamos a setear que siempre que se escriba o se genere un archivo lo haga con grupo administracion-Z_RW. Para esto existe un parámetro llamado `force_group` que hará que cada vez que se cree un archivo le fuerce ese grupo. Primero eliminemos el archivo que teníamos antes para evitar problemas:

```
root@samba-server:~# rm /data/discos/administracion-Z/upload.samba
```

Ahora ingresemos al archivo de configuración de Samba:

```
root@samba-server:~# vi /etc/samba/smb.conf
```

Y al final de todo, dentro de la sección `[administracion-Z]` podremos agregarle lo siguiente:

```
force group = administracion-Z_RW
root@samba-server:~# /etc/init.d/samba restart
```

Luego creemos un archivo utilizando a `aramis` que es quien tiene permisos de escritura sobre esta carpeta:

```
root@samba-server:~# su - aramis
aramis@samba-server:~$ touch samba.upload_aramis
```

Luego subiremos la información:

```
samba-server:~$ smbclient //172.16.0.138/administracion-Z -U aramis
smb: ¥> put samba.upload_aramis
smb: ¥> quit
```

Y revisemos los propietarios:

```
samba-server:~$ ls -l /data/discos/administracion-Z/
-rwxr--r-- 1 aramis administracion-Z_RW samba.upload_aramis
```

Pero todavía tenemos un inconveniente que es la máscara de creación. Como ven, lleva máscara 744 que nos llevará al mismo inconveniente, otros usuarios no podrán compartir información de este recurso, o mejor dicho, no podrán editar lo que se coloque en este recurso que es la idea de un disco compartido. Antes tenemos que entender el paralelismo entre las máscaras de Windows y las máscaras de Linux.

PERMISOS Y ATRIBUTOS DE GNU/LINUX VS DOS

DOS nunca fue pensado para ser un sistema multiusuario y en red; pero no así Unix. Consecuentemente, existe inconsistencias entre los dos filesystem que Samba debe proveer una solución. Uno de los problemas es como Unix y DOS manejan los permisos de archivos.

Tomemos como ejemplo el archivo que creamos anteriormente en el servidor:

```
samba-server:~$ ls -l /data/discos/administracion-Z/samba.upload_aramis
-rwxr--r-- 1 aramis administracion-Z_RW /data/discos/administracion-Z/samba.upload_aramis
```

Como vemos existen 9 bits que pueden estar en 1 o 0; pero en Windows solo tiene 4 bits que se utilizan con cualquier archivo:

- read only
- archive
- hidden
- system

Esto hace que los 4 bits de ejecución de Linux no estén presentes en un archivo en un disco Samba share... mejor dicho, están; pero no se utilizan. Sin embargo, los archivos de DOS tienen sus propios atributos que necesitan ser preservados cuando son guardados en un entorno Unix (archive, system, hidden). Samba puede preservarlos reutilizando el bit de execute. Sin embargo, el mapeo de estos bits tiene un efecto secundario: Si un usuario Windows guarda un archivo en un disco compartido Samba, y se ve en un equipo Linux con el comando `ls -al`, algunos bits ejecutables no significan lo que nosotros creemos que significan.

Existen 3 opciones de Samba que deciden como los bits son mapeados:

- map archive
- map system
- map hidden



Estas opciones mapean los atributos archive, system, y hidden al bit de ejecución de owner, group y others.

Estos se pueden agregar utilizando las siguientes líneas dentro del archivo de configuración, en la sección de configuración del datashare. Las líneas que activan este mapeo son (aún no las pongamos).