LINUX VOICE TUTORIAL

SAMBA 4: IMPLEMENT ACTIVE DIRECTORY DOMAIN SERVICES

JOHN LANE

Master your Windows domain from the comfortable familiarity of your Linux server.

WHY DO THIS?

- Administer Windows machines on a network without having to abandon your Linux working environment.
- Learn one of the most important features in Samba 4.

LV PRO TIP

Implementing Samba requires root privileges. **sudo -i** gives you a root prompt.

Samba is an open source implementation of the protocols for user and resource management in a Windows network. It allows Unix-like operating systems such as Linux and OS X to share files and printers, and to authenticate and manage users and resources in a Windows network.

The venerable version 3 series had long satisfied the file sharing needs of many Linux systems, until Microsoft introduced its Active Directory user and resource management infrastructure. But version 4 of Samba resolves this, because it is fully-compatible with it. In this tutorial, we'll install the Samba Version 4 server and configure it as an Active Directory Domain Controller. Up-to-date distros should have updated their Samba version, but you can always download the latest sources from the **samba.org** website. We'll use the "Trusty Tahr" Ubuntu Server, version 14.04, as it's a long term support release that includes Samba 4.1.6 in its repositories. This makes installation straightforward – as root:

```
$ apt-get install samba smbclient
```

We also installed **smbclient**, the command line Samba client. We'll use it to help test our server.

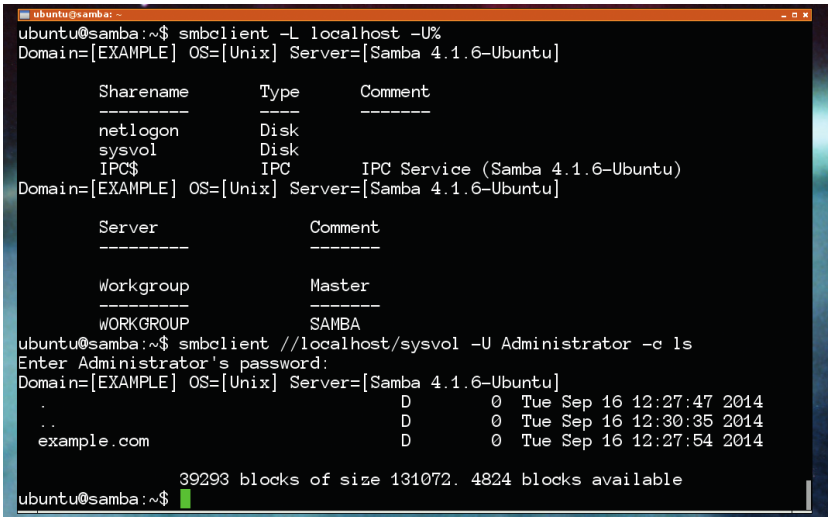
Ubuntu's Samba package automatically starts the daemons upon installation. We're about to reconfigure it, so stop them now:

```
$ stop smbd
$ stop nmbd
```

Samba's main administration tool, **samba-tool**, is used to provision (set up) a new domain controller. You need to remove the pre-installed default Samba configuration file before you begin otherwise provisioning will fail (it writes a new one and won't overwrite an existing one):

```
$ rm /etc/samba/smb.conf
```

A new server hasn't got much to share, but there's no harm in looking.



You should also ensure that your server is configured with a static IP address and has itself listed as its primary name server. If you need help configuring this, our Network Configuration box explains what to do.

Interactive provisioning prompts for you to enter the required information but offers default values that are usually acceptable. The first question asks for a Realm, which is the domain suffix that Active Directory will apply to all hosts that join the domain. The default value is the default search domain for your network, as defined in **/etc/resolv.conf** and converted to upper case letters (eg **EXAMPLE.COM**) and it's fine to accept this suggestion.

You will also be asked to choose a DNS Backend. Samba requires a DNS server and implements one internally if you accept the default **SAMBA_INTERNAL** option. This should be suitable for most uses but you can use an external BIND DNS server if you prefer.

The provisioning tool asks two questions that require non-default answers. You need to supply:

- The DNS Forwarder Address: the IP address of another DNS on your network, such as another name server defined in **/etc/resolv.conf**;
- An Administrator Password of your choosing that is suitably complex – it needs to have least eight characters containing three of these four kinds: lower-case letters, upper-case letters, digits and symbols. We'll use **"Pa$$w0rd"** in this tutorial; you should use something different.

Provisioning can be as simple as:

```
$ samba-tool domain provision --interactive
```

however, it's best to add some optional arguments to gain some additional benefits:

```
$ samba-tool domain provision --interactive --use-rfc2307
--use-xattrs=yes
```

The **--use-rfc2307** argument configures Active Directory so that it can store Unix user attributes, and this makes it possible to authenticate Linux users with Samba. The second argument allows Samba to support access control lists. These are lists of permissions that augment the basic **user**, **group** and **others** entitlements. Windows makes extensive use of them.

To support access control lists, the Linux kernel and any filesystem that you want to use with Samba

need to have extended attribute (abbreviated to 'xattr') support. You should be fine with the ext4 filesystem, but options for various other filesystems are explained at **https://wiki.samba.org/index.php/OS_Requirements**. You'll also need the **attr** and **acl** packages. Ubuntu 14.04 includes all of this by default.

You can start Samba when provisioning completes; the Ubuntu-specifc way to do this is to use Upstart:

```
$ start samba-ad-dc
```

but you can instead run the daemon directly, a distro-agnostic approach that is also useful when testing: to run it in the foreground with debug logging you can use:

```
$ samba -i -d 2 -M single
```

These, and many other, command line options are documented on the daemon's manual page (**man 8 samba**).

With Samba running, you can exercise the DNS to ensure it returns the expected results:

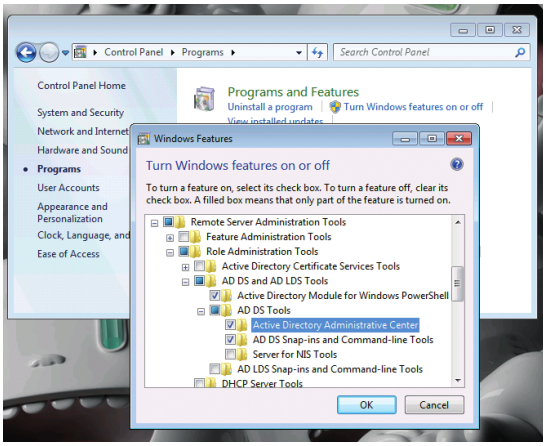
```
$ host -t SRV _ldap._tcp.example.com
_ldap._tcp.example.com has SRV record 0 100 389 samba.example.com.
$ host -t SRV _kerberos._udp.example.com
_kerberos._udp.example.com has SRV record 0 100 88 samba.example.com.
$ host -t A samba.example.com
samba.example.com has address 10.0.100.1
```

You may have seen the notification when the provisioning completed that "a Kerberos configuration suitable for Samba 4 has been generated". Kerberos is the authentication protocol used by Active Directory and the generated configuration allows you to interact with Samba's Kerberos services. Doing so is optional but useful for testing. If you want to use it, copy it into place and install the Kerberos client utilities:

```
$ cp /var/lib/samba/private/krb5.conf /etc
$ apt-get install krb5-user
```

You can then run some basic Kerberos tests (the Samba server needs to be running):

```
# kinit administrator@EXAMPLE.COM
Password for administrator@EXAMPLE.COM:
# klist
Ticket cache: FILE:/tmp/krb5cc_0
```



Installing RSAT is not enough: you must also use Turn Windows Features On Or Off to enable it.

What is Active Directory?

Active Directory, or its more complete and up-to-date name, Active Directory Domain Services, (ADDS) is a scalable, secure, and manageable infrastructure for user and resource management.

A server that provides ADDS has the ADDS 'Server Role' and is called a 'domain controller'. Its responsibilities include authentication and authorisation of users and computers in a Windows network, the assignment and enforcement of security policies and installing and updating software. The "directory" part refers to a listing of "objects". It's a database that is managed by a "Directory System Agent" (DSA) and can be accessed using the Lightweight Directory Access Protocol (LDAP); there are also ADSI, MAPI and "Security Accounts Manager" (SAM) interfaces. The objects are either "resources" or "security principals", the latter having unique "Security Identifiers" (SIDs). Unlike the earlier Windows NT domain controllers, it's possible for there to be multiple servers with the ADDS role, all accepting read/write operations and replicating changes to remain in sync.

ADDS uses Kerberos for authentication.

```
Default principal: administrator@EXAMPLE.COM

Valid starting     Expires            Service principal
16/09/14 12:42:07  16/09/14 22:42:07  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 17/09/14 12:41:56
```

We can use the Samba client tool to browse our domain's shares. We can list them and connect to them to see their contents (you'll need to enter the password that you chose during provisioning).

```
$ smbclient -L localhost -U%
$ smbclient //localhost/sysvol -U'Administrator%Pa$$w0rd' -c ls
```

Another way to access shares is to mount them using the cifs filesystem:

```
$ mount -t cifs -o username=Administrator,password='Pa$$w0rd' //samba/sysvol /mnt
```

LV PRO TIP

If you want to use less secure passwords: **samba-tool domain passwordsettings set --complexity=off**

Serving time

Participants in an Active Directory domain work best when they have synchronised time clocks because Active Directory uses Kerberos for authentication, and this is extremely time-sensitive. There is an allowed tolerance of five minutes and any more than this will result in denied access. It's also essential if you have multiple servers because directory replication relies on synchronised clocks. Implementing a time server will allow clients attempting to connect to our server to synchronise their clocks from it.

Microsoft uses an extension to the standard Network Time Protocol that uses signed timestamps. It calls this the "Windows Time Service". The standard **ntpd** time server can provide such times by having Samba sign its timestamps. Install the daemon from the repository:

```
$ apt-get install ntp
```

Modify the configuration file so that **ntpd** asks Samba to sign its timestamps. You need to define the socket where the signing agent listens and add a server restriction so that requests get signed by default. If you're using a virtual server, such as LXC, you can replace the whole **/etc/ntp.conf** with the following example, otherwise amend your existing configuration so that it includes the last two lines. Restart the daemon after making your changes (**service ntp restart**).

LV PRO TIP

Borked install? Just delete **/etc/samba/smb.conf** and **/var/lib/samba/private** and start over.

Network configuration

Your Samba server needs a static IP address and it should be configured to use Samba as its primary DNS name server. You can use a static IP configuration to achieve this by editing **/etc/network/interfaces.d/eth0.cfg** so that it reads like this:

```
auto eth0
iface eth0 inet static
address 10.0.100.1
netmask 255.0.0.0
gateway 10.0.0.138
dns-nameservers 10.0.100.1 10.0.0.138
dns-search example.com
```

You'll need to use values appropriate to your own network. Our server's interface is **eth0** but yours may be different and you should use your own domain name and an IP address appropriate to your network.

You can use DHCP if you prefer but you will need to make sure that your DHCP server always assigns the same IP address to your network interface. You can get your network's interface (MAC address) by doing:

```
$ cat /sys/class/net/eth0/address
```

You'll need to prepend the settings supplied by DHCP with the local DNS server entry and there are various ways to do this. One way on Ubuntu is to add it to **/etc/resolvconf/resolv.conf.d.head** like this:

```
nameserver 10.0.100.1
```

You should also set a host name in **/etc/hostname** and its fully-qualified domain name in **/etc/hosts**. We're using **samba.example.com** so our **/etc/hostname** file contains just the host name, like this:

```
samba
```

and our **/etc/hosts** file contains a line like this:

```
127.0.1.1 samba.example.com samba
```

The easiest way to ensure your network settings take effect is to reboot after making them so that the **\etc\resolv.conf** file that DNS relies on is updated. You can then confirm your settings:

```
$ hostname
samba
$ hostname -f
samba.example.com
$ cat /etc/resolv.conf
nameserver 10.0.100.1
nameserver 10.0.0.138
search example.com
```

```
server 127.127.1.0
fudge 127.127.1.0 stratum 12
ntpsigndsocket /var/lib/samba/ntp_signd/
restrict default mssntp
```

Our example uses **127.127.1.0** as a time server address. This is a pseudo-address that NTP recognises as its own local clock and synchronises with itself. This is sufficient inside a virtual server whose clock is controlled by the VPS host.

The **ntpsigndsocket** entry defines the path to the directory where Samba places the socket file on which it will listen for signing requests. The path is determined by Samba's configuration and you can confirm the correct path with:

```
$ samba-tool testparm --verbose --suppress-prompt | grep "ntp signd socket directory"
        ntp signd socket directory = /var/lib/samba/ntp_signd
```

Samba creates the socket directory but you should ensure that it is writeable by the **ntpd** daemon, which usually runs as **ntp:ntp**. You should change the directory's group to match:

```
$ chgrp ntp /var/lib/samba/ntp_signd
```

Unfortunately there is no tool to test NTP authentication from Linux but we can do so when we connect our first Windows client to our Samba server. The following examples assume that you have a clean install of Windows 7, and bear in mind that you can't join a domain from the Starter or Home editions although you'll still be able to access shares.

There are a couple of prerequisites before a client can join the domain. The first is that it must use the Samba server's DNS. The second requirement is for its clock to be reasonably consistent with the Samba server, say within a few seconds, otherwise errors may be reported that bear no relationship to the real problem and you will not be able to authenticate. The Windows time service will keep the clocks synchronised once the client becomes a domain member. We'll assume you know how to make these tweaks or know a Windows tech who does.

Now, to add the client to the domain, go to Start > Computer > Right-click > Properties > Change Settings. This will display the System Properties dialog, where you should click on the Change button and then select Domain in the Member Of section and enter the Samba domain name before pressing 'OK'. This should request the administrator account credentials (the username is 'Administrator' and password is 'Pa$$w0rd' if you've followed our example settings). It should finish by welcoming you to the domain and asking you to restart the computer.

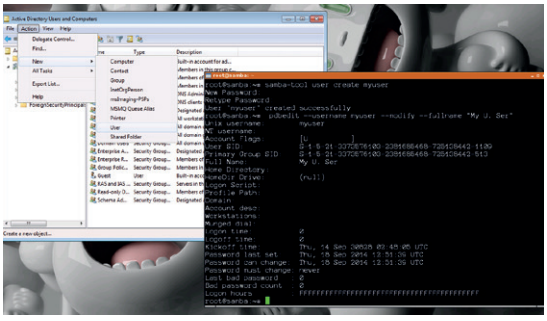
Log in as your domain administrator, (EXAMPLE\Administrator), when Windows restarts. You can now test NTP. Open a command prompt window as the Administrator (click the Start button, type **cmd** and then right-click the **cmd** icon that appears in the search results to select Run As Administrator) and then:

```
C:\> w32tm /resync
Sending resync command to local computer
The command completed successfully.
```

So, we now have Active Directory Domain Services and have joined a client to the domain. What does that give us? Well, we can now use Windows tools to administer our domain, but you need to download the Remote Server Administration Tools and install them. Do this while you're still logged in to your Windows desktop – see **http://bit.ly/ms-rsat**.

The Remote Server Administration Tools include a tool called Active Directory Users And Computers that you can use for your admin tasks. Run this, as an administrator, via the Start button: search for **dsa.msc** (this is the name of the relevant executable file that you need to run). The Action menu lists the various administrative actions that you can perform, such as adding a new user.

You can also perform these tasks using the Samba command line tools if you prefer that way of doing things. The Samba administration utility is called



Whatever your preference, you can get your admin done: you can use the native Windows tools or the various Linux command line alternatives.

LV PRO TIP

Reload Samba's config without restarting: **smbcontrol all reload-config**. Sanity check it with **testparm**.

LV PRO TIP

ntp_signd is a compile-time option. If signed requests do not work you may need to rebuild **ntpd** from source. This isn't the case with Ubuntu 14.04.

samba-tool, and you can use it to add users like this:

```
$ samba-tool user create myuser
```

This creates a user but doesn't enrich it with supplementary data that can be stored in Active Directory, such as their name and phone number, but you can use the **pdbedit** command line tool for that:

```
$ pdbedit --username myuser --modify --fullname "My User"
```

You can edit common user attributes with **pdbedit** but there are many more attributes in the directory that you can access. You'll need a basic grasp of how LDAP stores data and you'll need the LDAP Database Tools to access it. Install the tools and try some queries:

```
$ apt-get install ldb-tools
$ ldbsearch -H /var/lib/samba/private/sam.ldb -b CN=myuser,CN
=Users,DC=example,DC=com
$ ldbsearch -H /var/lib/samba/private/sam.ldb -b
CN=Users,DC=example,DC=com samaccountname=myuser
```

The first argument points at Samba's database – your Active Directory. The second argument is the Distinguished Name (DN) to search within (a DN is what uniquely identifies a record in LDAP and the base DN specifies where to start the search). What follows the arguments is an expression that selects records from the database and fields from those records. If the expression is omitted then everything beneath the base DN is returned. See **man ldapsearch** for more.

Use your preferred method to try adding a user now, we'll make use of **myuser** in the following examples. If you need to edit your user's record then **ldbedit** gives you direct edit access to the directory. Be careful not to alter any internal Active Directory data. You can edit a user like this:

```
$ ldbedit -H /var/lib/samba/private/sam.ldb -b
CN=Users,DC=example,DC=com samaccountname=myuser
```

Linux login

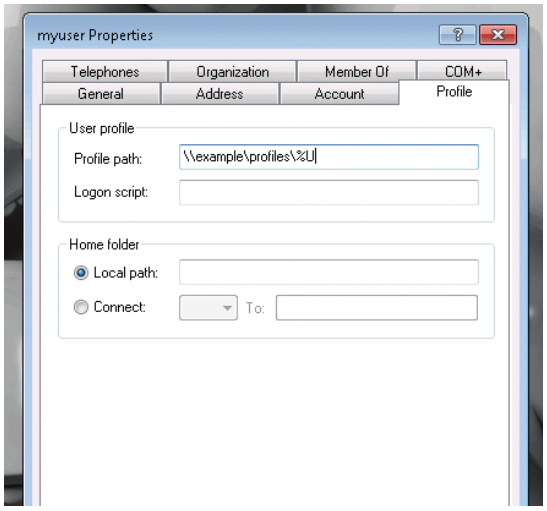
We recommended adding a **--use-rfc2307** option when provisioning the Samba server. RFC2307 is an internet standard that Active Directory implements so that it can store Unix attributes like usernames and passwords in a standard way. The provisioning option instructs Samba to do similarly and this allows us to use Samba to authenticate users that log in to our Linux machines. Microsoft's Active Directory implementation calls this "Identity Management for UNIX". If you want to authenticate users in this way, their computers need **winbind**, a daemon that looks up usernames and passwords in Active Directory. You need to install it, along with libraries that link it into the authentication process:

```
$ apt-get winbind libnss-winbind libpam-winbind
```

NSS is the Name Service Switch and you need to configure it to use **winbind** as a data source by adding it after the options already in place. Our modified Ubuntu **/etc/nsswitch.conf** looks like this:

```
passwd:     compat winbind
group:      compat winbind
```

You can test these using **getent passwd** and **getent group**, and you can look up your user with **id**:



Roaming profiles link to the **[profiles]** share configured in **smb.conf**. The **\%U** in the path will be replaced with the username.

```
$ id myuser
uid=3000021(EXAMPLE\myuser) gid=100(users)
groups=100(users)
```

Domain users have high-numbered UIDs that are assigned by Active Directory. You can modify this (or any other LDAP attribute) using **ldbedit** but they're kept separately from the main directory. You need a user's Security Identifier, or SID, to find them. The SID is another way that Active Directory uniquely identifies a user. The commands you need are:

```
$ wbinfo --name-to-sid myuser
S-1-5-21-3373576103-2381685468-725138442-1109
SID_USER (1)
$ ldbedit -H /var/lib/samba/private/idmap.ldb cn=S-1-5-21-
3373576103-2381685468-725138442-1109
```

The field that you need to change is **xidNumber**; you can set this to the desired **uid** value. You only really need to do this when moving existing users from **/etc/passwd** into the directory.

You can try logging in as the user you created earlier, for example:

```
$ ssh myuser@my_linux_box
```

When in Roam...

File and print sharing works exactly as it does when Samba is used in the classic, non-Active Directory, way by writing stanzas in **smb.conf**. One thing that a domain controller adds to this is Roaming Profiles. This feature enables your domain users to log in to Windows clients and download their user profile directory. Think about your users' habits before enabling roaming profiles. Because they are downloaded and uploaded inefficiently, users storing large amounts of data in their profile can put undue pressure on your Samba server.

There's much more to Active Directory than we've covered here, but you should be able to get your first server up and running and save yourself from one more proprietary server.

John Lane provides technical solutions to business problems. He has yet to find anything that Linux can't solve.

LV PRO TIP

If you have Apparmor on your server, check that its configuration allows access to the Samba socket (it does on Ubuntu 14.04). See **/etc/apparmor.d/usr.sbin.ntpd**.

LV PRO TIP

From a Windows command line, use **ipconfig /all** to check network settings such as DNS.