« **Increasing Productivity through Clean Air** **Mail Server Security**

NBD and PXE Booting on Debian

I have a Xen server that I use for testing which is fairly lightly loaded. I considered making it diskless to save some electricity use (which also means heat dissipation in summer) and also some noise.

The first step is to setup a PXE server. **This is reasonably well documented in the Debian Administration article on setting up PXE [1]**. Basicall the DHCP configuration needs to include the line "**filename "pxelinux.0";**" to tell the workstation the name of the file to download. This file is downloaded from a TFTP server, so you need to install one (I chose the **tftpd-hpa** package). The **pxelinux.0** is provided by the **syslinux-common** package, I believe that the Debian Administration article errs in not mentioning this fact, they recommend using **wget** to download it which means tha there is no verification of the file contents.

It appears that the way PXE works is that you are expected to have a directory named **pxelinux.cfg** under the root of the TFTP tree which then contains PXE configuration files. The Debian Administration article gives an example of using a file named **default** but you can also name a file for th MAC address of the workstation, a number which appears to be a GUUID for the workstation, and the IP address in hexadecimal (if that doesn't exist then it will be truncated one nibble at a time, so 10.10.10.100 will result in searches for **0A0A0A64**, **0A0A0A6**, … **0**). That's what my HP test machine does.

The Debian Administration article shows how to configure PXE for installing Debian. But I wasn't interested in that, I wanted to convert a system that i running as a regular workstation to be diskless. The first step in doing this is to install the **nbd-client** package which results in rebuilding the initrd to have support for diskless operation. Then you have to install the **nbd-server** package on the file server. The documentation for this package suggests that it is designed to serve regular files as block devices, but it appears to work OK with LVM devices. Adding an export section such as the following to **/etc/nbd-server/config** causes an LV to be exported via NBD:

```
[export]
exportname = /dev/vg0/workstation0
port = 12345
authfile = /etc/nbd-server/allow
listenaddr = 192.168.0.1
```

Then it's just a matter of copying the filesystem from the hard drive to the LV that is used for NBD. I piped tar through ssh to copy the root filesystem c a running system. But I could have copied the block device or used **debootstrap** to create a new image from scratch.

NBD has an interesting design in that it exports block devices (which can be backed by files or real block devices) to a particular set of IP addresses and uses a particular TCP port for the export. So if you have two NFS exports from one server you might have 192.168.0.1:/home and 192.168.0.1:/data as exports but if you have two NBD devices you might have 192.168.0.1,12345 and 192.168.0.1,12346. This could be considered t be very sensible or utterly wrong.

The final thing to do is to setup a PXE configuration file. I put the following in a file named **pxelinux.cfg/default**, if I was going to deploy this seriously would replace **default** with the IP address of the system.

```
DEFAULT lenny_i386

LABEL lenny_i386
    kernel lenny/vmlinuz-2.6.26-2-686
    append selinux=1 nbdroot=192.168.0.1,12345 initrd=lenny/initrd.img-2.6.26-2-686 root=/dev/nbd0 ip=dhcp --
```

The only things I needed to change in the image that I'm booting after transferring it from the hard drive is **/etc/fstab** and the network configuration **/etc/network/interfaces** – obviously if the network start scripts change the IP address of the workstation and thus make the root filesystem unavailable then things will break.

Wouter has some more background information on this [2]. He recommends using partitioned NBDs, that's a matter of opinion, if I was going to use this in production I would use two NBDs, one for the root filesystem and another for LVM which would be used for everything else. I really like to be able to create snapshots and to change the size of LVs at run-time.
The down-side of LVM is that it can be really inconvenient to access LVM volumes when not running the machine that owns them – there is no suppo for using an LV as a PV (IE nested LVM) or for having two VGs with the same name running on the same machine.

Wouter also seems to be planning to write Debian Installer support for using NBD as a target. This would be a nice feature.

Now the next thing is to use Xen. Xen makes it a little more exciting because instead of having two essential files to be loaded (the kernel and the initrd/initramfs) you have three (the Xen kernel plus the other two). So we need to chain to a different boot loader. **The Gentoo Wiki has good information on installing this [3]**.

The summary is that you need to chain the **mboot.c32** loader from PXE which is then used to load the Xen kernel, the Linux kernel, and the initrd. Below is an example that I attempted. This loaded the correct files, booted Xen, and then hung. I didn't investigate the cause.

```
DEFAULT mboot.c32 xen-3.2-1-i386.gz dom0_mem=258048 --- lenny/vmlinuz-2.6.26-2-xen-686 ro xencons=tty console=tty0 selinux=1
root=/dev/nbd0 ip=dhcp nbdroot=192.168.0.1,12345 --- lenny/initrd.img-2.6.26-2-xen-686
```

The configuration for **mboot.c32** is particularly ugly. I think it would be better to have a replacement PXE loader which includes the mboot support.

I ended up deciding not to use NBD for the machine in question, the process of upgrading kernels (which is not uncommon on a test machine) would be made more difficult by the process of copying them to the tftp server, I guess I could write a script to rsync them. I had a problem with the system shutdown scripts killing the **nbd-client** process and hanging the system, I guess I could patch the shutdown scripts to ignore certain processes (this would be a good feature) or I could use SE Linux policy to prevent **nbd-client** from being killed by any domain other than **sysadm_t**. But generally it seemed to be more effort than saving 7W of power is worth.

- [1] **http://www.debian-administration.org/articles/478**
- [2] **http://grep.be/blog/en/computer/nbd/root-on-nbd**

- [3] **http://www.gentoo-wiki.info/Xen#Alternative:_PXELinux**

Related posts:

1. **installing Xen domU on Debian Etch** I have just been installing a Xen domU on Debian...
2. **MySQL security in Debian** Currently there is a problem with the MySQL default install...
3. **Ethernet Bonding on Debian Etch** I have previously blogged about Ethernet bonding on Red Hat...
4. **booting from USB for security** Sune Vuorela asks about how to secure important data such...
5. **Installing a Red Hat based DomU on a Debian Dom0** The first step is to copy /images/xen/vmlinuz and /images/xen/initrd.img from...

August 23rd, 2009 | Tags: Most Popular | Category: Networking

1 comment to NBD and PXE Booting on Debian

wouter verhelst
August 23, 2009 at 22:56

Hi Russell,

Two notes:

First, the nbd-client initscript already writes the PID of the nbd-client process that serves the root filesystem to a file under the directory /lib/init /rw/sendsigs.omit.d/, which causes the initscript to *not* kill your nbd-client process from under your nose when shutting down the system. In lenny, it doesn't yet work if you have root=/dev/vg0/something rather than /dev/nbd0 or some such, but other than that, it should work. However, that does require that you have the nbd-client package installed, and that you have a line saying KILLALL=false in /etc/nbd-client.

Second, if the documentation gave you the impression that you can only export normal files (and not block devices), then the documentation needs fixing. Exporting block devices is perfectly supported, and is not a problem. I did recently commit a change to the upstream git repository (see **http://git.grep.be/gitweb?p=nbd.git;a=commit;h=cd3279e5a3441b833bc4a75d8228fc6160ef14c9**) that effected a change to the documentation to make this more explicit; could you review comment?