Walkthrough: Creating your own network appliance

Introduction

With the massive growth of GNU/Linux and other Open Source Operating Systems, it's never been easier to build your own network appliance.

I've been asked by a client to build a network device providing the following functionality;

- Virtual Private Networking (VPN)
- Web Caching
- Web Content Filtering
- User Manual Filing
- Network Monitoring (security and/or performance)

Hardware

I want the hardware to be rack mountable and to absorb as little space as possible. In order to achieve this I've opted to use an old Servgate Edgeforce Plus firewall. Not only is the equipment well within my size specifications, but the internals are essentially a Pentium III PC with 256MB of RAM.

This should be more than sufficient for our needs.

The equipment also includes a hardware cryptographic accelaration card for VPN, although I'm not currently certain whether we can configure OpenVPN to use it.

We'll be installing the base system onto a 8GB Harddrive. I've chosen this disk purely because it was spare and so minimises costs for the customer. Any harddrive of a reasonable size will do (the hardware also supports flash cards, I may look at booting the system from this at a later date)

Software

The base OS will be Gentoo Linux. We'll be installing binary packages from the Live CD/DVD, but the really determined could then opt to re-compile and optimise everything once the system is set up. Given the services that we'll be running, performance is key, but a full recompile will take a while so it's a choice best left until the end.

Requirement	***Software***
Virtual Private Networking	OpenVPN
Web Caching	Squid
Web Content Filtering	DansGuardian
User Manual Filing	DocLibrary
Network Monitoring	Snort and Others

There are also a few key software requirements that I have added to the project;

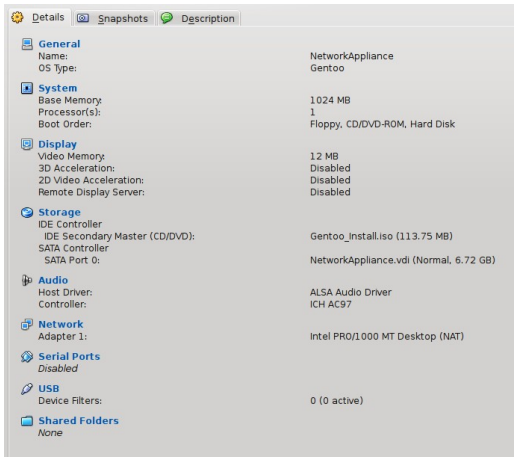
Requirement	***Software***
Remote Configuration	Webmin & OpenSSH
File Upload	ProFTPD
File Sharing	Samba (Will be disabled until needed)

All software used is Open Source and available free of charge. Where possible, we'll be utilising the software in Gentoo's repositories instead of manually installing each package.

Preparation

The hardware has a serial port based console output. I've not got a compatible monitor, and am not inclined to wait for a Pentium III with 256MB of RAM to complete the installation.

Instead I've opted to complete the initial install in a Virtual Machine. So I've created a 6.75GB disk image in ~~Sun~~ Oracle VirtualBox OSE and will be installing to that. It also carries the benefit of not having to find a blank CD to burn the Gentoo image to!



Base Installation Image

To begin, we need to install Gentoo as the base OS for our network appliance (We're using the minimal install disc). So once the Installation Disk Image has finished downloading, we need to load our VirtualBox Appliance and boot from the CD Image.

There's no obvious reason why we'd need to boot the kernel with custom options, so simply hitting enter at the prompt sets us on our way (obviously YMMV).

Cryptographic Accelerator Support

Whilst waiting for the VM to boot, it seems wise to conduct some research into whether or not the hardware crypto card will be supported. The card is based on a Broadcom 5820 chipset.

A quick search suggests that it may be supported, but as ever, there's no concrete confirmation. As the hardware may be supported, it seems wise to defer installation of OpenVPN until the base system is operating on the target hardware.

Installing Gentoo - Preparation

We are using the minimal install CD, so there's no GUI available. Older versions of Gentoo had a LiveCD installer, but I can't seem to find it in the latest version (2010.01). I've always preferred the minimal install route anyway!

When asked enter the value corresponding to your keymap (40 for the UK)

Networking

We need to ascertain whether the system has detected our network card;

ping -c 3 www.google.com

Rather annoyingly, mine wasn't. VirtualBox uses a virtual Intel 82540EM Gigabit controller, so I simply needed to run the following commands;

modprobe e100
ifconfig eth0 up
dhcpcd eth0

After which I was able to ping google.

Great! Now we can proceed;

Preparing the Harddrive

We currently have an empty Harddisk image, before we can even consider an install we need to create and format some partitions.

- Boot - 32MB
- Swap - 512MB
- Root Partition – Rest of Disk

So we need to run

fdisk /dev/sda
n
p
1
[Enter]
+32M
a
1

We've now created our boot partition and made it bootable (the a).

Next we need to create our swap partition, so (still in fdisk we run)

n
p
2
[Enter]
+512M

Although this has created the partition we need to set it as swap;

t
2
82

Now we have a boot partition and a swap partition. The final task is to create the main system partition;

n
p
3
[Enter]
[Enter]

Our final partition now occupies the rest of the disk, leaving us with a partition table looking like this (use *p* to display)

```
Command (m for help): p

Disk /dev/sda: 7218 MB, 7218397184 bytes
255 heads, 63 sectors/track, 877 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe2b3e8c9

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1           5       40131   83  Linux
/dev/sda2               6          71      530145   82  Linux swap / Solaris
/dev/sda3              72         877     6474195   83  Linux
```

We now need to save the new layout to disk so enter

w

Formatting Partitions

Without a filesystem our partitions are still largely useless. So lets begin by formatting both the boot and the root partitions as ext2 (Note: We could use ext3 but performance is key and the extra journalling isn't really necessary for this type of appliance.)

mke2fs /dev/sda1
mke2fs /dev/sda3

Our partitions are both now formatted. The swap partition simply needs initialising and activating;

mkswap /dev/sda2
swapon /dev/sda2

We've now not only created our swap partition, but we've lent the benefit of it to our Virtual machine.

Mounting the Partitions

In order to install to the partitions, we need to make the accessible to the current system. At a later point we'll be chrooting into them.

mount /dev/sda3 /mnt/gentoo/
mkdir /mnt/gentoo/boot
mount /dev/sda1 /mnt/gentoo/boot/
cd /mnt/gentoo

We're now going to proceed with the installation itself.

Beginning the Installation

We're now cd'd into /mnt/gentoo. We're going to use the default method of installation by retrieving a stage tarball from the internet;

links http://www.gentoo.org/main/en/mirrors.xml

Use the keyboard to navigate to a mirror closest to you. Once you've highlighted the link press enter to follow it. I used the bytemark http link.

You should see a list of directories, choose *releases → x86 → Autobuilds → Current-x86-install*
Select the Stage3 i686 tarball and press *d* to download. Hit enter when prompted for a filename.

Once that has completed press *q* to quite links.

Unpack the stage 3 tarball onto the system

tar jxpvf stage3-i686-20101116.tar.bz2

Once this has finished (it takes a while), the next job is to install the application manager.

Install Portage

Portage is a fantastically powerful application, we use it to search for and install software from the Gentoo repositories. Debian users will be familiar with apt-get, Portage is simply the Gentoo equivalent.

For those who did not get on with *links*, I've some bad news: We're about to use it again!

links http://www.gentoo.org/main/en/mirrors.xml

Choose a mirror (I'm using the Bytemark http again) and select snapshots.
Choose the most recent portage snapshot (should be portage-latest) and press *d* to download. Once again press enter to accept the default file name.

Once the file has downloaded, press *q* to exit links.

Next we'll be extracting the archive in order to install portage;

tar jxvf portage-latest.tar.bz2 -C /mnt/gentoo/usr

(Note the capital C)

The tarball will extract and we now have an installation of portage. This should be the last time we need to use links, at least for quite a while!

Optimize the System

At this point, we could set optimisation flags for gcc in /mnt/gentoo/etc/make.conf. However, these will make very little difference to a system as small and light as the one we are building. If you wish to set optimisation flags, the Gentoo Handbook has examples that you can use.

Chrooting into the new system

We need to be chrooted into the new system in order to effect the next stage of installation.

First, we need to tell the system which mirror to download from. This can make a very very big difference in the time required to complete installation. We'll use the automated tool rather than messing about with links again

mirrorselect -o -i >> /mnt/gentoo/etc/make.conf
mirrorselect -o -r -i >> /mnt/gentoo/etc/make.conf

Locate the mirror closest to you and select it by using the space bar. Then press Enter to exit mirrorselect. Repeat for the second command as well

This has set the mirror for both http and rsync.

Next we need to ensure that we make a copy of our DNS resolution information, otherwise the new system won't be able to resolve hostnames and URLs

cat /etc/resolv.conf > /mnt/gentoo/etc/resolv.conf

Final Steps

All that remains before chrooting is to give the subsystem access to both /proc and /dev

mount -t proc none /mnt/gentoo/proc
mount -o bind /dev/ /mnt/gentoo/dev

Chroot

Now we need to chroot into our new system;

chroot /mnt/gentoo /bin/bash
env-update
source /etc/profile

We're in the new system! Admittedly we still have a lot of work left to do, but doesn't it feel **good?**

Download Updates

Momentary excitement aside, it's time to update the ebuilds within the portage tree. To do so run the following;

emerge --sync

You'll quickly become familiar with emerge, it's the main section of portage that we'll be using. It does absolutely everything from installing to updating.

On the offchance you should get a warning saying that a newer version of portage is available (I did), run the following

emerge --oneshot portage

Now with an up-to-date portage installation, we can continue to set our USE flags.

USE Flags

First, a little explanation: USE Flags define how our applications are compiled. Other distributions will release pre-compiled packages with support for everything including the kitchen sink™, with a resulting increase in memory footprint. We need something much leaner, so we'll be excluding support for things that we don't need.

The use flags and their explanation are below;

Flag	*Prefix*	*Explanation*
X	-	Our equipment will not have a display and there's no requirement for X forwarding. There's therefore no need for X support in our applications.
kde	-	As with X, there's no requirement for kde support
gnome	-	See above
alsa	-	Alsa is the Advanced Linux Sound Architecture. Our equipment does not have a sound card, nor is it likely to require one.
cgi		Allows support for Common Gateway Interface scripts. This will be required by doclibrary
ipv6		Adds support for IPV6, which we're all told is coming soon

Now we need to translate this into an entry in /etc/make.conf. First we load the file in nano

nano /etc/make.conf

and add the line *USE=”ipv6 cgi -alsa -gnome -kde -X”* Before pressing *Ctrl-X* followed by *[Enter]* to exit and save.

Configure the Kernel

The Kernel is the most important aspect of the Operating System. Without it nothing else works, it is essentially the translator sat between the hardware and userspace.

Lets begin by making sure our timezone is set correctly;

cp /usr/share/zoneinfo/GMT /etc/localtime

Next, we need to install our new kernel. There is a choice, but I generally use gentoo-sources.

emerge gentoo-sources

Once this has completed, the kernel source has finished installing. We simply need to configure and compile the kernel now. The bad news is, I've no idea exactly what hardware support we need for the Edgeforce unit, so it'll all be an educated guess!

To start with we need to navigate to the kernel source directory

cd /usr/src/linux

Then we need to launch the configuration program

make menuconfig

At this point it becomes a little difficult to note what I do, so please bear with me. To compile something as a module press *m*, to compile it into the kernel press *space* until a star appears in the box (will also toggle through M and not compiled). To move up a level press *Esc*.

Processor type and Features →
Processor Family → Pentium Pro (Press Space)
Up a Level
File Systems →
Second Extended FS Support (Press Space)
{Exit and Save}

The default options seem fine for the time being, we can always recompile later if needs be. The main thing is the essentials are configured so our system will be able to boot.

Next we need to compile the kernel;

make && make modules_install

Note: *If the system is unbootable when we place it in the target hardware, to fix it we simply need to boot from the LiveCD and chroot in to reconfigure the kernel. We're not going to be faced with a full re-install!!!*

Next we need to install the kernel image into our boot partition.

cp arch/i386/boot/BzImage /boot/kernel-2.6.35-gentoo-r12

Configuring Modules

In the last step we compiled the kernel and along with it some modules, we now need to tell the system which of these modules to load at boot.

Although we didn't manually set any modules there may have been some pre-configured that we will want. In order to check we run the following command

find /lib/modules/2.6.35-gentoo-r12/ -type f -iname '.o' or -iname '*.ko' | less*

I've not got any that need adding, but if you do open */etc/modules.autoload.d/kernel-2.6* in a text editor and simply list the module names (one per line)

Configure the System

For some reason Gentoo provides an invalid */etc/fstab* file so we need to create our own to ensure that our partitions mount correctly.

nano /etc/fstab

We need to delete the old entries so that the file contains the following;

```
/dev/sda1       /boot           ext2    noauto,defaults         1 2
/dev/sda3       /               ext2    noatime                 0 1
/dev/sda2       none            swap    sw                      0 0
proc            /proc           proc    defaults                0 0

shm             /dev/shm        tmpfs   nodev,nosuid,noexec     0 0
```

We don't need to add a cdrom entry as the target kit doesn't have an optical drive.

Network Settings

We now need to set a few things network wise;

Hostname

What do we want to call our hardware? I'll be calling mine Ouroboross

We need to edit the hostname file;

nano /etc/conf.d/hostname

And set HOSTNAME to the hostname you've chosen - We'll worry about the DNS domain later.

Network Address

We'll begin by setting our system to use DHCP. We can reconfigure this once the system is installed on the target hardware.

emerge dhcpcd
nano /etc/conf.d/net

If we had one network card we'd be updating the file just to contain the first of the following lines. The target hardware, however, has three NIC's so we'll set each to use DHCP (especially as we have no idea which is which!)

config_eth0=("dhcp")
config_eth1=("dhcp")
config_eth2=("dhcp")

Although we've configured the NIC's to use DHCP, we need to tell the system to start them at boot. We'll add the init script for eth0 to the default boot sequence. If we need the others at a later date we'll add the scripts for those.

rc-update add net.eth0 default

Networking is now configured! We're almost ready to go.

Basic Configuration

We now need to set the root password (as an example we'll use P455w0rd)

passwd
P455w0rd
P455w0rd

And the root password is set!

Now we need to set our keyboard layout

nano /etc/conf.d/keymaps

Change the value of KEYMAP to uk

Additional Applications

If we were following the Gentoo handbook, we'd now be installing system logging. We don't need it at the moment, so we're going to skip over that. We can always install it later!

Similarly, we don't need a cron daemon at the moment.

We do, however, want the following;

- Webmin
- Dansguardian
- Squid
- Openssh

So, we'll ask Portage to perform the install;

emerge webmin dansguardian squid openssh

After a little while, the packages are installed.

We need to set an administrator password for Webmin, so we'll run the following command (assuming a password of 1234567)

/etc/init.d/webmin start
/usr/libexec/webmin/changepass.pl /etc/webmin root 1234567
/etc/init.d/webmin stop

We can now proceed with configuring the bootloader.

Note: We'll install and configure the other software later.

Configure the Bootloader

We'll be using Grub as Lilo is more than a little dated nowadays.

emerge grub

Next we need to edit the configuration file

nano /boot/grub/grub.conf

Enter the following (you may need to delete the existing entries)

```
# Which entry to boot by default. 0 is the first, 1 the second etc.
default 0
# How many seconds to wait before the default boots.
# As this is a network appliance we want this to be nice and short
timeout 5

title Gentoo Linux 2.6.35-r12
# Partition where the kernel image is located
# /dev/sda1 would be hd0,0
#/dev/sda2 would be hd0,1 etc.
root (hd0,0)
kernel /boot/kernel-2.6.35-gentoo-r12 root=/dev/sda3

title Gentoo Linux 2.6.35-r12 (rescue)
# Partition where the kernel image is located
root (hd0,0)
kernel /boot/kernel-2.6.35-gentoo-r12 root=/dev/sda3 init=/bin/bb
```

Save and exit

Sadly, because we are chrooted we can't just run *grub-install* so we need to make an extra change first.

cat /proc/mounts > /etc/mtab

Now we can run grub-install and install the Bootloader to the Master Boot Record (MBR)

grub-install --no-floppy /dev/sda

Et Voila! Grub is installed. We now need to exit the chroot and reboot the system;

```
exit
cd /
reboot
```

Tell VirtualBox to unmount the CD image:

Devices → CD/DVD Devices → Unmount CD/DVD Device

You may need to wait until the system has finished rebooting before you're able to unmount the CD image.

If all goes well, our new system should boot!!!! We've reached a major milestone, there's only a little configuration left to do.

Our New System

If all's gone well, you should be looking at the console login screen. Username is root and the password is whatever you set it as earlier (I used *P455w0rd*)

Before we even think about moving the system to the target hardware, there's a few steps left to complete;

Add the SSH Daemon and our other services to the default runlevel

rc-update add sshd default
rc-update add webmin default

We should then be able to configure Dansguardian and Squid from our Webmin installation. To test our webmin installation we will need to start it

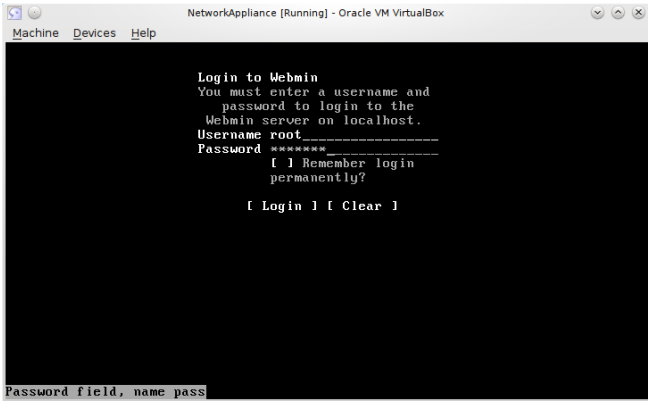
/etc/init.d/webmin start

To access it, we need a web browser. Time to install *links*!!!

emerge links

Then to access Webmin we use

links https://localhost:10000



At the login screen we enter the username root and the password we set earlier (I used 1234567).

The system may or may not grant you access, links isn't exactly a supported browser! The main thing is that webmin is listening on port 10000 as expected.

The next step is to convert the VirtualBox image into a raw disk image and dump it onto our harddrive.

First we need to exit the virtual machine by running the command *halt*. The we need to instruct VirtualBox to convert the disk image for us;

VboxManage clonehd --format RAW NetworkAppliance.vdi NetworkAppliance.img

Note: If the above didn't work for you, see the note at the bottom of this document

Dump to Disk

Next connect the harddrive to your PC/Laptop using a USB to IDE adaptor. I'll assume that it's detected as /dev/sdb

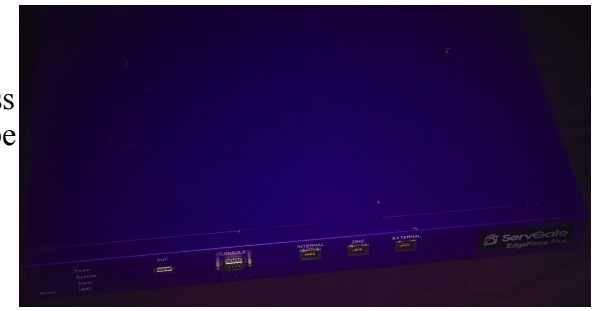
You'll need to be root for the next step, whether that's by using *su* or *sudo*.

cat NetworkAppliance.img > /dev/sdb

Once the process is completed, we can install our harddrive into the target hardware. Before hand, read Note for the Impatient in case it applies to you!

Boot the Hardware

Once we've installed the harddrive into the hardware, we need to connect it to the network and power it up. With a little bit of luck, it'll boot and grab an IP address using DHCP. If our router is playing nice, we should be able to address the hardware using the DNS name we configured earlier;



ping -c 3 ouroboross

If not, we'll need to check the DHCP table of our router. If we can't find an entry, then we need to think about which ethernet port we used, only eth0 is configured to use DHCP. Assuming we started on the port labelled 'Internal' we need to try each of the other two until we find one that works.

I'll assume that this wasn't necessary for the rest of this walkthrough.

Note: If you're not planning on configuring the hardware immediately, you need to read the Security Note below.

Next Steps

See document two for the steps following this guide

Alternative to VBoxManage

For some reason, VboxManage failed to convert my VDI. Although the command returned no errors, it sat at 0% perpetually and refused to respond to commands.

It could be an issue with my install, or a bug in VirtualBox, but I've not got the luxury of time to investigate it. Instead, I booted the Gentoo Virtual Machine from the installation CD again.

I then made sure I had space for an 8 Gigabyte image on one of my other systems and issued the following command (once I'd brought the network up);

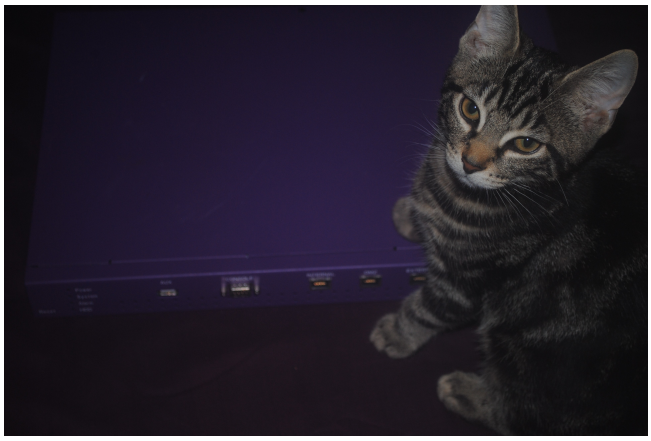
cat /dev/sda | ssh ben@kryten 'cat > /mnt/exthd/NetworkAppliance.img'

The command will take a little while to run, then we just need to move it back to our host system and dump the image to disk.

Security Note

If you've installed the harddrive into the target hardware, but are not intending on configuring it immediately, be aware that a variety of security settings will still be at their default settings. Depending on how paranoid you are, this is very bad as the system could easily be tampered with by anyone able to gain access to the system (which could be a lot if you've plugged it into the network without configuring.)

It's therefore very important that you take the appropriate security precautions when storing the system prior to configuration. At the very least, you should be securing the hardware in a locked location. If, like me, you're truly paranoid you could arrange a security guard like the one pictured below;



For the Impatient

When I began this project, I was unable to find my IDE to USB adaptor and so couldn't make an image of the target harddrive. So I created a disk image using VirtualBox with the intention of dumping the system to disc once I found the adaptor.

Although this will work, there is an extra step. Because the image was not created from a physical drive, the Master Boot Record may be in a different location on disc.

So, once the image has been dumped to disc we need to grab the image back off and install grub into the MBR.

So assuming the drive has been detected as sdb.

cat /dev/sdb > Appliance_Image2.raw
VBoxManage convertfromraw Appliance_Image2.raw Appliance_Image2.vdi

Once these have completed, fire up VirtualBox and define the new VDI as the harddrive. Then use the CD installer to boot the virtual machine. Follow the Chroot steps followed by Installing the Bootloader (Just the final two steps). Once this has completed you can unmount the drive and exit VirtualBox.

We then just need to convert the VDI back to a RAW disc image. In order to do so;

VBoxManage internals converttoraw Appliance_Image2.vdi Appliance_Image_Final.raw

We can now proceed by dumping the image to disc;

cat Appliance_Image_Final.raw > /dev/sdb

We're ready to proceed with installing the hardware.