“A Linux Security Gateway"

Autor: **Nacho Ruiz Ramón**
Plan de estudios: **Enginyeria Tècnica d’Informàtica de Sistemes**
Area: **Plataforma GNU/Linux**
Consultor: **Joaquín López Sánchez-Montañes**
Tutor: **Josep Soler Masó**
Fecha: **Junio 2016**

FITXA DEL TREBALL FINAL

Títol del treball:	
Nom de l'autor:	
Nom del consultor:	
Data de lliurament (mm/aaaa):	*MM/AAAA*
Àrea del Treball Final:	
Titulació:	*Pla d'Estudis de l'Estudiant*

Resum del Treball (màxim 250 paraules):

Distribución personalizada basada en Debian pensada para un entorno empresarial, cuya función principal es ser es un firewall de red, también conocido como cortafuegos. Pero además proporciona otros servicios considerados básicos como serian servidor de *dhcp*, servidor de *ntp*, acceso *vpn* externo, servidor de *dns*, *webproxy* y compartición de carpetas y ficheros en red. Esto permite tener en un único dispositivo lo que de otra manera se conseguiría con varios dispositivos independientes. También ofrece funcionamiento en cluster, pudiendo ser configurado para comunicarse con un segundo servidor y ofrecer opciones de alta disponibilidad. Así, se gana en facilidad de administración, reducción de los costes, rapidez en la resolución de problemas, etc. Al ofrecer prácticamente funciones de red los requerimientos de procesador y memoria serán muy básicos y el coste económico de este servidor será muy bajo, pudiendo funcionar prácticamente en cualquier ordenador del mercado. El único requisito como imprescindible seria que el servidor tuviese dos dispositivos físicos de red. Tiene como principal característica la facilidad y rapidez de instalación. La instalación en un servidor se realiza en pocos minutos a partir de una simple imagen iso ya se a partir de un dispositivo USB o un cd-rom, y la configuración en entornos sencillos es cuestión de otros pocos minutos más.

Abstract (in English, 250 words or less):

Custom Debian based distribution designed for a business environment, whose main function is to be a network firewall. Also it provides other essential services as *dhcp* server, *ntp* server, external *vpn* access, *dns* server, *webproxy* and sharing folders and files on the network. This allows a single device in what otherwise would be achieved with several separate devices. It also offers cluster operation and can be configured to communicate with a second server and provide high availability options. So you gain in ease of management, cost reduction, speed troubleshooting, etc. By offering mainly network functions the processor and memory requirements are very basic and the economic cost of this server will be very low and can run on virtually any computer available on the market. The only requirement will be that the server will need two physical network devices. Its main characteristic the ease and speed of installation. The installation is performed on a server in minutes from a single image iso already from a USB device or a CD-ROM, and simple configuration other environments is a matter of few minutes.

Paraules clau (entre 4 i 8):

Linux Debian Distribución Firewall Servidor Router

índice

1.1. Contexto y justificación del Trabajo

En la mayoría de empresas donde trabajaremos como Ingenieros de Sistemas, tendremos que gestionar los servicios de acceso a Internet que tenga contratados. Esto supone tener que configurar y conocer en profundidad uno o más dispositivos que nos definirán a que servicios el personal de la empresa puede tener acceso, cuales no, cuales son accesibles desde Internet, etc. Este servidor será lo que conocemos generalmente como cortafuegos o *firewall* y mediante una seria de reglas definiremos básicamente que servicios son accesibles y cuales no. Además de esta funcionalidad inicial se definirán una serie de servicios que se han considerado fundamentales y que para la gran mayoría de empresas serán de uso cotidiano e imprescindibles.

Actualmente existen multitud de soluciones comerciales que nos cubren y facilitan esta gestión pero estos servidores suelen ser caros y van asociados a exclusivos y largos contratos de mantenimiento, lo que hace complicado una vez instalado dentro de la empresa buscar alternativas mas económicas y asequibles.

1.2. Objetivos del Trabajo

El objetivo de este Trabajo es, a partir de la distribución Debian[1], realizar una instalación inicial del sistema operativo base y a partir de este punto ir describiendo todo el proceso de instalación, configuración y mantenimiento de los diferentes servicios que ofrecerá este servidor. También se mostrará como se construye esta distribución personalizada para poder ser distribuida y mantenida de manera fácil incluso por un equipo de personas.

Al final tendremos un servidor que nos podrá ofrecer varios servicios, principalmente, pueda realizar tareas de *firewall*, servidor de *dhcp*, servidor de *ntp*, acceso *vpn* externo, servidor de *dns*, *webproxy* y compartición de carpetas en red. Este servidor personalizado también ofrecerá opciones de *clustering*, pudiendo ser configurado para comunicarse con un segundo servidor y ofrecer opciones de alta disponibilidad (cuando por cualquier razón el servidor principal dejase de dar servicio el segundo que estaría en *standby* adquiría las funciones del principal, todo ello de forma totalmente automática). Al ofrecer prácticamente funciones de red los requerimientos de procesador y memoria serán muy básicos y el coste económico de este servidor será muy bajo, pudiendo funcionar prácticamente en cualquier ordenador aún siendo antiguo. El único requisito como imprescindible sería que el servidor tuviese dos dispositivos físicos de red.

Esta distribución que hemos personalizado la transformaremos en una distribución propia que puede ser instalada con los diferentes componentes configurados en cualquier servidor compatible o incluso como es posible instalarla en un entorno virtual. Esta distribución recibirá el nombre de “Naxtaro”.

1.3. Enfoque y método seguido

A partir de la distribución de Linux Debian[1] que instalaremos en *Virtualbox[2]*, procederemos a construir nuestra distribución personalizada que llamaremos “Naxtaro”. Usaremos el propio software de *Virtualbox* para realizar las pruebas e ir instalando las imágenes que obtengamos de la distribución Debian inicial.

1.4. Planificación del Trabajo

Para el desarrollo de este proyecto, se seguirán una serie de fases que coincidirían en fechas con las distintas PACs que tenemos que entregar. Hemos considerado para la planificación los días que tenemos de plazo para la entrega de cada PAC.

1. **PAC1**. Plan de trabajo - 22 días
   Realizaremos las tareas necesarias para definir y plantear los objetivos que se quieren cubrir dentro del proyecto (el presente documento).

2. **PAC2**. Diseño e implementación - 21 días (+8 días)
   En esta parte definiremos que requerimientos de hardware necesitaremos, que distribución en concreto hemos elegido y comenzaremos la instalación del sistema operativo base y los componentes que creemos serán útiles para nuestro servidor.

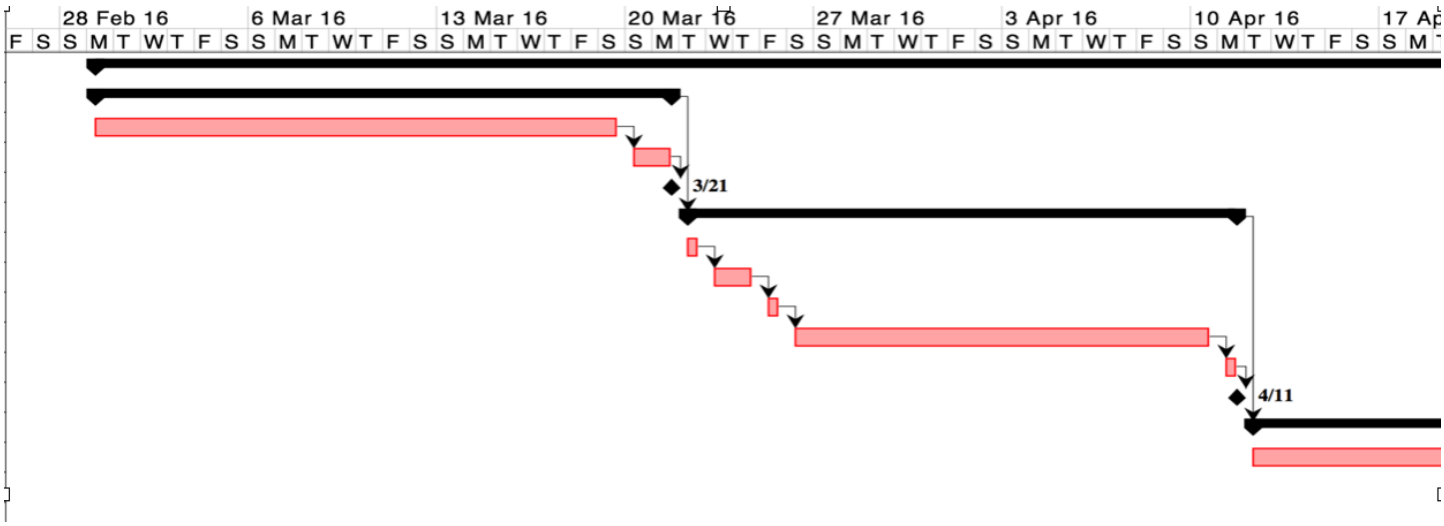
3. **PAC3**. Resultados y análisis - 28 días (-8 días)
   Se ejecutarán pruebas de verificación y tests de la parte anterior, crearemos nuestra propia distribución y la instalaremos en entornos virtuales. En este periodo también configuraremos los clientes de diferentes dispositivos para comprobar los accesos desde el exterior (móviles, *tablets*, portátiles, etc). También se demostrará como se configuraría el servidor para conectar dos o mas sedes de una empresa internamente y como configurar un segundo servidor para formar un *cluster* de alta disponibilidad. Finalmente se configurarán servicios de monitorización, gráficas estadísticas, etc

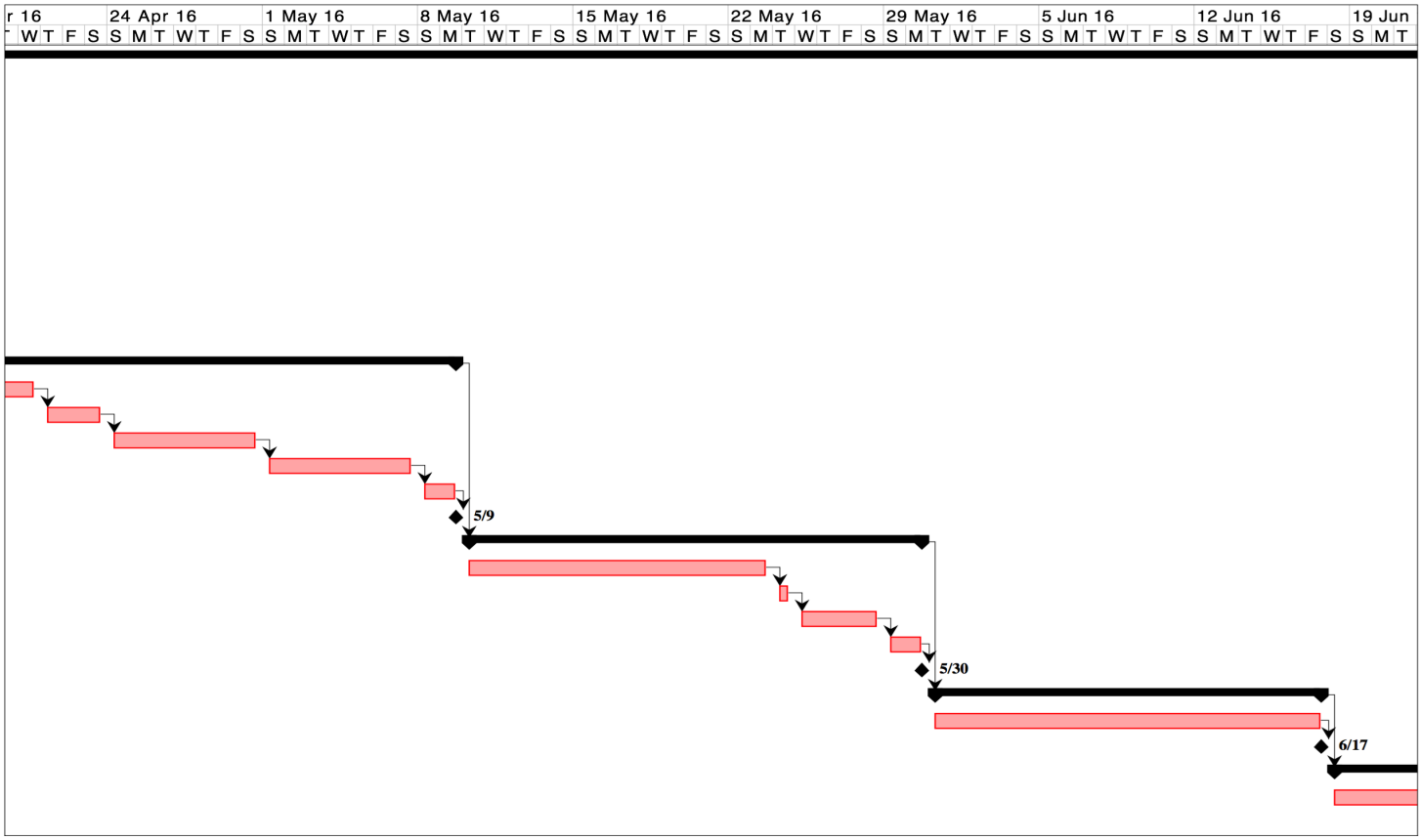
4. **PAC4**. Entrega memoria - 21 días
   Realizaremos y presentaremos la versión prácticamente final de todo el trabajo, conclusiones finales, incluyendo un manual de uso y configuración del servidor.

5. **Memoria final y producto** - 18 días
   En esta parte básicamente revisaremos toda la memoria, corregiremos posibles errores, etc.

6. **Video presentación** - 7 días
   Producción del video-presentación virtual, mediante una serie de *slides* presentaremos el proyecto de manera visual.

		Name	Duration	Start	Finish	Predecessors
1		**TFC GNU/Linux**	**117 days?**	**2/29/16 8:00 AM**	**6/24/16 5:00 PM**	
2		**PAC1**	**22 days**	**2/29/16 8:00 AM**	**3/21/16 5:00 PM**	
3		Elaboración documento plan de trabajo	20 days	2/29/16 8:00 AM	3/19/16 5:00 PM	
4		Revision PAC1	2 days	3/20/16 8:00 AM	3/21/16 5:00 PM	3
5		Entrega PAC1	0 days	3/21/16 5:00 PM	3/21/16 5:00 PM	4
6		**PAC2. Diseño e implementación**	**21 days?**	**3/22/16 8:00 AM**	**4/11/16 5:00 PM**	**2**
7		Elección de hardware y componentes	1 day	3/22/16 8:00 AM	3/22/16 5:00 PM	
8		Elección distribucion base	2 days?	3/23/16 8:00 AM	3/24/16 5:00 PM	7
9		Instalación distribución base	1 day	3/25/16 8:00 AM	3/25/16 5:00 PM	8
10		Implementación de funcionalidades	16 days	3/26/16 8:00 AM	4/10/16 5:00 PM	9
11		Revisión PAC2	1 day	4/11/16 8:00 AM	4/11/16 5:00 PM	10
12		Entrega PAC2	0 days	4/11/16 5:00 PM	4/11/16 5:00 PM	11
13		**PAC3. Resultados y análisis**	**28 days**	**4/12/16 8:00 AM**	**5/9/16 5:00 PM**	**6**
14		Pruebas, tests	9 days	4/12/16 8:00 AM	4/20/16 5:00 PM	
15		Creacion de distribución propia	3 days	4/21/16 8:00 AM	4/23/16 5:00 PM	14
16		Instalación en entornos virtuales	7 days	4/24/16 8:00 AM	4/30/16 5:00 PM	15
17		Configuración y tests en alta disponibilid..	7 days	5/1/16 8:00 AM	5/7/16 5:00 PM	16
18		Revisión PAC3	2 days	5/8/16 8:00 AM	5/9/16 5:00 PM	17
19		Entrega PAC3	0 days	5/9/16 5:00 PM	5/9/16 5:00 PM	18
20		**PAC4. Entrega memoria**	**21 days**	**5/10/16 8:00 AM**	**5/30/16 5:00 PM**	**13**
21		Documentación y presentación	14 days	5/10/16 8:00 AM	5/23/16 5:00 PM	
22		Producción y envio de la distribución	1 day	5/24/16 8:00 AM	5/24/16 5:00 PM	21
23		Manual de uso y configuración	4 days	5/25/16 8:00 AM	5/28/16 5:00 PM	22
24		Revisión PAC4	2 days	5/29/16 8:00 AM	5/30/16 5:00 PM	23
25		Entrega PAC4	0 days	5/30/16 5:00 PM	5/30/16 5:00 PM	24
26		**Memoria final y producto**	**18 days**	**5/31/16 8:00 AM**	**6/17/16 5:00 PM**	**20**
27		Revisión final memoria	18 days	5/31/16 8:00 AM	6/17/16 5:00 PM	
28		Entrega final	0 days	6/17/16 5:00 PM	6/17/16 5:00 PM	27
29		**Video presentación**	**7 days**	**6/18/16 8:00 AM**	**6/24/16 5:00 PM**	**26**
30		Realización del video presentación	7 days	6/18/16 8:00 AM	6/24/16 5:00 PM	
31		Entrega video	0 days	6/24/16 5:00 PM	6/24/16 5:00 PM	30





1.5. Breve sumario de los productos obtenidos

Distribución propia basada en Debian Linux[1], con funcionalidades principales de *firewall*, acceso remoto por *vpn*, servidor de *dns*, servidor horario y compartición de ficheros por red.

1.6. Breve descripción de los otros capítulos de la memoria

En lo siguientes capítulos se planteará una planificación del trabajo inicial dividida en varias fases principales. Que herramientas y entornos necesitaremos para empezar a trabajar, los requerimientos que necesitaremos como será un sistema Linux base donde se configurará todo el proceso de automatización de la creación de la distribución, conocer a fondo la herramienta y sus diferentes opciones que hace posible todo el proceso, se describirán en detalle las características y funciones de cada servicio, como se instalan y cómo se configuran en este entorno. A continuación mediante el desarrollo de un programa a medida se obtendrán las variables necesarias para completar la instalación de la distribución. Finalmente se llegará a la parte donde se ha de probar que todo esto funciona y analizar si los resultados son los esperados o no, así como una serie de conclusiones finales de valoración sobre el Trabajo realizado.

2. Tecnología empleada

He elegido la distribución Debian por tratarse de un Sistema Operativo Linux robusto, fácil de instalar y de administrar gracias al amplio catálogo de software disponible en forma de paquetes que se gestionan de manera muy sencilla (unos 43.000 paquetes) y desde la línea de comandos. Además se gestiona automáticamente que software depende de que otros y se instalará cualquier dependencia que sea necesaria, además de sugerir otro software relacionado con el que queremos instalar. A parte todo este software está disponible en formato código fuente, que podremos estudiar o modificar según nuestras necesidades.

Se encuentra disponible en múltiples arquitecturas y varios idiomas, es muy estable y también tenemos actualizaciones y parches a nivel de seguridad a la orden del día.

Utiliza tres modelos de distribución, la *unstable*, la *testing* y la *stable*[4]. La *unstable* como su propio nombre indica es una distribución muy inestable, solamente recomendada para poder probar nuevas versiones de software y sus componentes (para hacernos una idea esta distribución se actualiza cada 6 horas!). Este nuevo software, una vez probado y si supera una serie de controles y pruebas pasa a la siguiente distribución, la *testing*, se vuelve a probar todo y finalmente pasa a la versión *stable*, que es la más robusta y consistente de todas. Es la que usaremos para elaborar este Proyecto. En concreto la ultima versión estable disponible, la 8.4, que ha sido liberada recientemente, el día 2 de Abril.

Se financia principalmente a partir de donaciones y a través de socios o *partners*, que ceden parte de su infraestructura o servidores para apoyar a Debian.

En realidad este Trabajo se podría realizar en cualquiera de las variantes existentes de Debian, como por ejemplo Ubuntu, Knoppix, Kanotix, etc ya que las aplicaciones utilizadas también están disponibles en todas ellas y ademas la gestión e instalación de los componentes seria prácticamente igual.

Cualquier duda o problema que tengamos la podemos plantear en más de las 280 listas de discusión que tiene disponibles de forma oficial, de hecho me he subscrito a varias para llevar a cabo este Trabajo, siendo la mas útil la de “debian live”[5] ya que es la utilidad que usaremos para la construcción de nuestra distribución.

Por otra parte, también usaremos *VirtualBox* como plataforma de pruebas y tests y poder desarrollar nuestra distribución hasta llegar a una versión final. Se trata de un software multiplataforma de virtualización. Es posible instalarlo en cualquier ordenador con procesador AMD o Intel, con Sistema Operativo Windows, Mac, Linux o Solaris, ampliando sus capacidades ya que podemos instalar varios sistemas operativos diferentes y que funcionen todos a la vez.

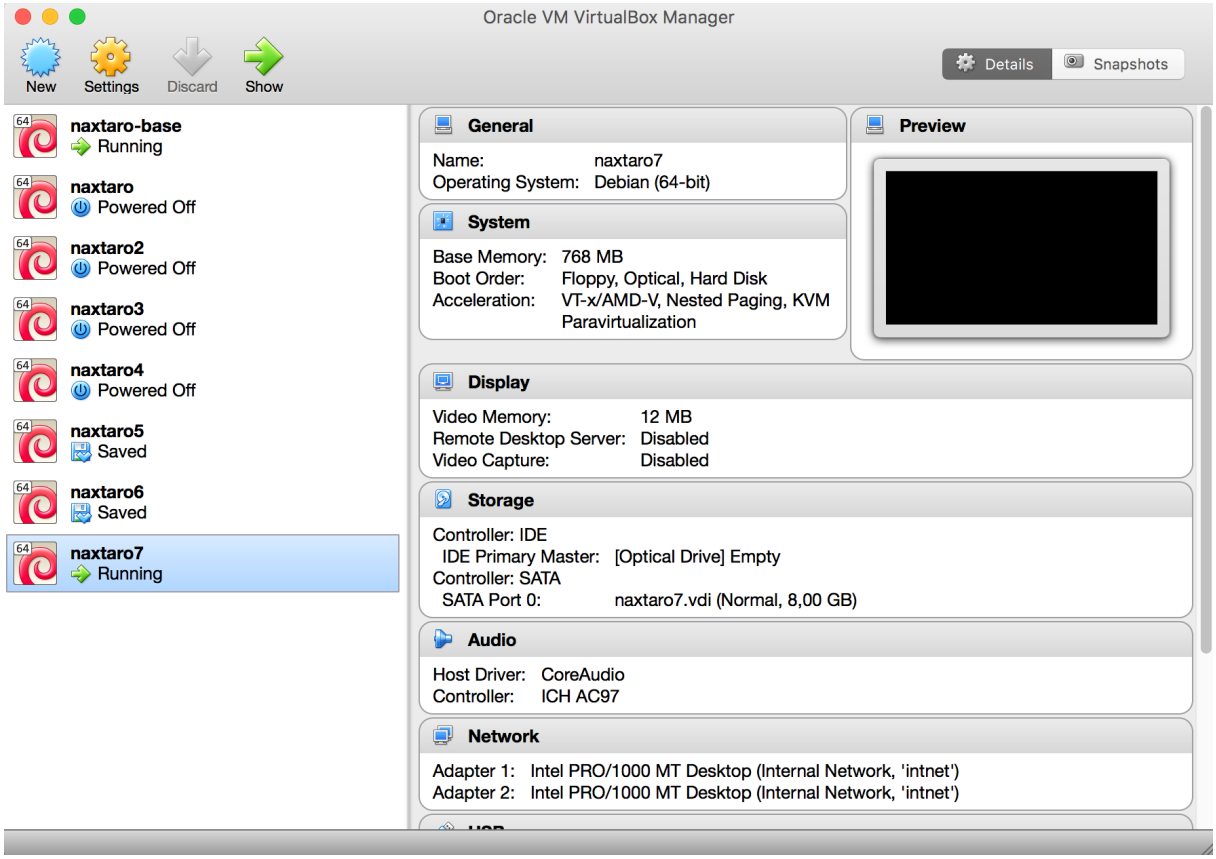
Por ejemplo es posible instalar en un ordenador Mac una maquina virtual Windows 10, en una máquina Windows instalar un Unix Solaris, etc. Es factible instalar todas las máquinas virtuales que queramos, la única limitación sera la memoria y disco físico de la maquina principal.

Es precisamente esta flexibilidad y facilidad de uso la que nos será muy útil para poder llevar a cabo la gran cantidad de pruebas que necesitará la creación del Proyecto.

Instalaremos la última versión disponible, la 5.0.20 para ordenadores con sistema operativo OS X, que es mi caso.

Como plataforma hardware se usará un ordenador portátil de la marca Apple, modelo MacBook Pro, fabricado a mediados del 2014, con 1 procesador de 4 cores a 2,2Ghz de velocidad, con 16Gb de memoria Ram y unos 500Gb de espacio de disco duro.

Finalmente también probaremos la instalación en un entorno virtual de servidor *Xen*[6] versión 4.4.



3. Funcionalidades de Naxtaro Linux

Las funcionalidades principales que incorporara la distribución serán

- Servicio de SSH
- Servicio de Firewall
- Servicio de apt-cache
- Servicio de MySQL
- Servicio de DHCP
- Servicio de DNS
- Servicio de VPN y site2site
- Servicio de NTP
- Servicio de Webproxy
- Servicio de Samba
- Servicio de Alta Disponibilidad

He considerado estos servicios como los básicos y que podrían ser imprescindibles para cualquier servidor que ha de prestar servicio en una empresa de tipo medio.

A partir de estas funcionalidades obviamente se pueden ir quitando o añadiendo nuevas siguiendo la metodología y los estándares que se describen en el presente documento.

4. Instalación del servidor base

Para realizar el proceso de construcción de la distribución necesitaremos un primer servidor “base” donde instalaremos el software y los componentes necesarios. Este servidor se usara únicamente para este propósito, por lo que solamente tendrá instalado lo necesario para tal función. A este servidor lo bautizaré como “naxtaro-base”.

4.1. Instalación de Debian base en *Virtualbox*

Antes de empezar a trabajar con *Virtualbox*, procederemos a descargar la versión escogida de Debian, de todas las disponibles, en concreto me interesa la versión “network install”, al ser la más ligera y apropiada ya que podremos elegir que componentes necesitaremos y no tendremos software innecesario instalado. Esta versión está disponible desde la url https://www.debian.org/CD/netinst/ y el nombre del fichero es `debian-8.4.0-amd64-netinst.iso`

Una vez tenemos en nuestro ordenador local el fichero *iso*, instalaremos la última versión disponible de Virtualbox desde www.virtualbox.org. Ejecutamos el programa y seleccionamos New, y crearemos nuestra máquina virtual con los datos:

Name: naxtaro-base
Type: Linux
Version: Debian (64-bit)

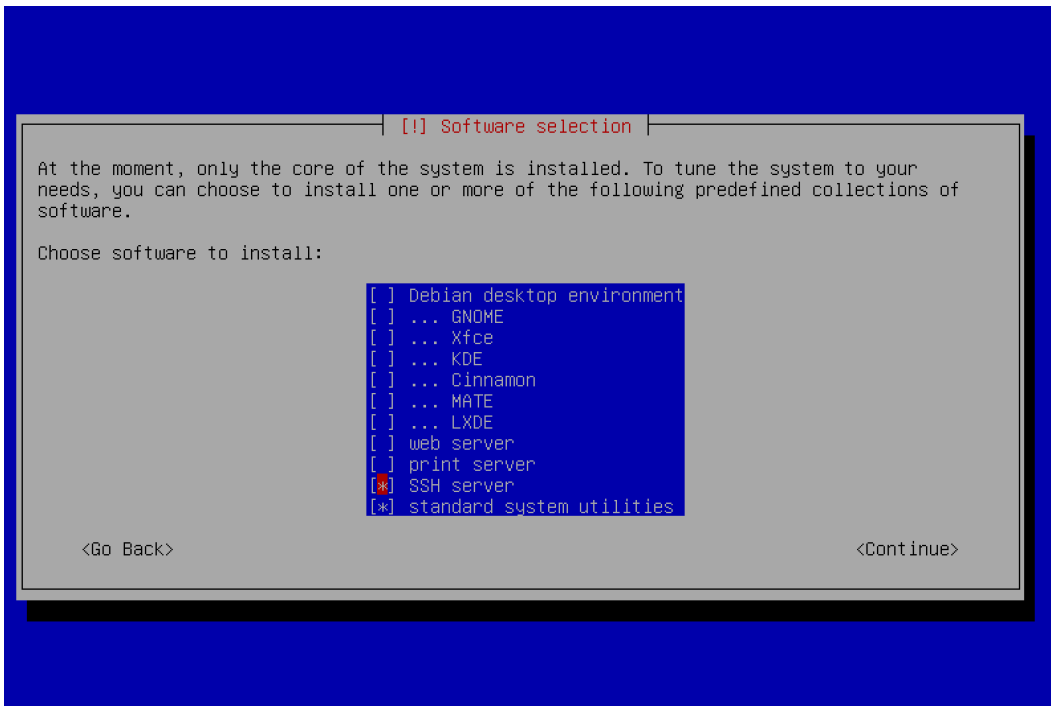
Memory Size: 768MB que nos mostrara por default es correcto, *Create a virtual hard disk now*, la opción recomendada de 8gb esta correcta, el tipo VDI (*virtualbox disk image*), opción *Dynamically allocated*, y finalmente *Create.*

En este punto procedemos seleccionando esta nueva máquina virtual y clickando el botón Start. Nos aparecerá un cuadro de diálogo donde seleccionaremos la imagen *iso* que tenemos en local.

Acto seguido aparecerá la pantalla inicial de instalación de Debian,

Ejecutamos Install, a partir de ahi el programa de instalación nos preguntará una serie de cuestiones básicas para poder instalar el sistema operativo. Este proceso no tendrá mayor complicación que seguir los pasos:

- asignar un password para el usuario privilegiado *root*
- crear un user genérico (crearemos un usuario llamado “naxtaro”)
- elegir lenguaje, tipo de teclado y zona horaria (elegiremos idioma inglés y zona horaria de Europe/Madrid)
- particionar el disco duro (no haremos particiones, no es necesario)
- escoger un mirror cercano (ftp.es.debian.org)
- En el apartado de software selection, solamente seleccionaremos SSH server y Standard system utilities:



- Finalizara la instalación y reiniciamos el servidor.
- Probaremos la conexión conectándonos mediante ssh.
- instalamos algunos paquetes básicos como *joe* (editor de textos), *screen* (permite recuperar una sesión después de una desconexión), etc.

4.2. Utilidad para desarrollar Naxtaro Linux

La aplicación escogida para el desarrollo será la conocida como “Live Systems Project”[7]. Esta aplicación en concreto es la que usan desde Debian para generar sus distribuciones Live (https://www.debian.org/CD/live/).
La aplicación *live-build* es una colección de scripts y de comandos para construir sistemas *live* o instalables ya que lleva incorporada la utilidad *debinstall* que es la usada por Debian para realizar la instalación de su sistema. Esta característica *live* nos permitirá poder probar ágilmente la distribución ya que no necesitaremos tenerla que instalar una y otra vez para ir verificando y corrigiendo cambios. Los comandos principales de la utilidad “lb” son:

lb config : Responsable de la inicialización del directorio de configuración donde residirá el sistema que queremos construir. Creará el esqueleto inicial de ficheros de configuración.
lb build : Responsable de iniciar la creación del sistema. Leerá la configuración del directorio “config” y procederá con una serie de comandos que construirán el sistema que queremos.
lb clean : Responsable de la limpieza del directorio de configuración.

En concreto incorpora una utilidad principal llamada “lb” que desde la línea de comandos nos permitirá poder configurar todo lo que necesitaremos para construir la distribución. Necesitaremos instalar en naxtaro-base los siguientes paquetes:

```
root@naxtaro-base:~# apt-get install live-build debootstrap
root@naxtaro-base:~# apt-get install dosfstools genisoimage mtools
parted syslinux uuid-runtime squashfs-tools mtd-utils
```

Adicionalmente durante este proceso constatamos que el proceso de creación del fichero imagen iso de la distribución, descarga cada vez los paquetes necesarios desde un servidor mirror de Debian, ftp.debian.org, añadiendo un tiempo extra en el proceso, por lo que vemos necesario utilizar algún sistema de *mirror* que nos permita tener una copia en el servidor base de los paquetes que una y otra vez necesitaremos en las numerosas pruebas que iremos realizando. Optamos por el aplicativo `apt-cacher-ng`[8].

```
root@naxtaro-base:~#  apt-get install apt-cacher-ng
```

Hay que cambiar la dirección de los repositorios por uno mas próximo en el fichero `/etc/apt-cacher-ng/backends_debian`
de `http://ftp.de.debian.org/debian/`
por `http://ftp.es.debian.org/debian/`
Reiniciamos el servicio para que los cambios tengan efecto:

```
root@naxtaro-base:~#  /etc/init.d/apt-cacher-ng restart
```

4.3. Configuración de la utilidad Live Build

El entorno base esta preparado para empezar la construcción de la primera versión de Naxtaro Linux, creamos un directorio llamado naxtaro

```
root@naxtaro-base:~# mkdir naxtaro; cd naxtaro
```

Ejecutamos el comando inicial de inicialización de nuestro futuro sistema, `lb config`

```
root@naxtaro-base:~/naxtaro# lb config
[2016-05-02 18:23:06] lb config
P: Creating config tree for a debian/jessie/amd64 system
P: Symlinking hooks...
```

Comprobamos como se han creado tres subdirectorios, “auto”, “config” y “local”, dentro del primero creamos un fichero llamado “config” que es donde indicaremos que parámetros queremos añadir al proceso de construcción,

```
#!/bin/sh
lb config noauto \
     --binary-images iso \
     --debian-installer live \
     --distribution jessie \
     --archive-areas "main contrib non-free" \
     --bootappend-live "boot=live config username=naxtaro hostname=naxtaro" \
     --mirror-bootstrap http://localhost:3142/debian/ \
     --mirror-binary http://localhost:3142/debian/ \
     "${@}"
```

(**Nota**: el símbolo especial "\" en los ficheros indica que el comando y sus argumentos forman parte de una única línea pero por motivos de espacio y facilitar la lectura hemos de escribirla en más de una).

Al tener todas las posibles opciones dentro de este único fichero nos aseguramos que cada vez que invoquemos al comando van a estar todas incluidas y también las tenemos en una sola ubicación controlada.

Los parámetros que nos interesan para la creación de la distribución son:

- parámetro especial *noauto,* utilizado para evitar entrar en un problema de recursividad infinita, es decir, hacer que solamente se pueda hacer una llamada al comando simultánea.
- parámetro "—binary-images iso", configuramos la opción de creación de una imagen del tipo estándar iso, la cual se puede instalar luego en un cd, en un dispositivo usb, en una maquina virtual, etc.,
- parámetro "—debian-installer live", opción interesante que es construir la imagen en formato live, esto nos sera muy util ya que podremos ejecutar y probar la distribución sin tener que instarla, por lo que solamente al arrancarla ya podemos comprobar que los componentes, la configuración y lo que hayamos incluido en ella se encuentran instalados correctamente,
- parámetro "--distribution jessie", nos interesa el tipo de distribución "Debian stable" también conocida con el sobrenombre de "jessie",
- parámetro "—archive-areas "main contrib non-free", seleccionamos el tipo de paquetes que necesitaremos,
- parámetro "--bootappend-live "boot=live config username=naxtaro hostname=naxtaro"", diferentes parametros que anadimos a la version live the la distribución, creamos el usuario local sin privilegios llamado "naxtaro" y definimos el nombre del host como "naxtaro"
- parámetros "--mirror-bootstrap http://localhost:3142/debian/" y "--mirror-binary http://localhost:3142/debian/", para que utilice como *mirror* de los paquetes de software la dirección del servidor local que previamente hemos instalado.

Por defecto la aplicación *livebuild* nos creara una distribución con la misma arquitectura que la que tiene la distribución origen, en nuestro caso *amd64*, si nos interesase crear una distribución para una arquitectura diferente, lo

tendríamos que especificar mediante el parámetro correspondiente, por ejemplo para una arquitectura *i386*, añadir al fichero “config” la opción:

```
--architectures i386
```

El siguiente fichero dentro del subdirectorio “auto” se llamará “build”, que será el que se lea cuando ejecutemos el comando “lb build”, con el contenido:

```
#!/bin/sh
set -e
lb build noauto "${@}" 2>&1 | tee build.log
```

se incluye la opción “noauto” que al igual que el anterior comprobará que sólo podamos hacer una llamada al comando simultánea, ”${@}” por si incluimos más parámetros, y con redirección de la *standard output (stdout)*, 1 y *standard error (stderr)*, 2, a un fichero llamado *build.log* donde se registrará todo el proceso de construcción.

Por último creamos un fichero llamado “clean” que sera justamente el que ejecute cuando invoquemos el comando “lb clean” con el contenido:

```
#!/bin/sh
set -e
lb clean noauto "${@}"
rm -f config/binary config/bootstrap config/chroot \
config/common config/source
rm -f build.log
```

Con la ya comentada opción “noauto”, y el borrado de los ficheros de configuración iniciales y el que registra el proceso.

El proceso de construcción se divide en distintas etapas, con diferentes personalizaciones aplicadas en secuencia en cada uno. La primera etapa inicial que se ejecuta es la etapa de arranque, llamada *bootstrap*. En esta fase se construye dentro del directorio *chroot* los paquetes necesarios para la creación del sistema. La segunda fase es la fase *chroot*, que completa la fase anterior con los paquetes extra que hemos definido para personalizar la distribución. La fase final es la llamada *binary*, que construye la imagen de inicio usando los contenidos del directorio *chroot* de la fase anterior para construir el sistema de ficheros raíz, que incluye el instalador (*debinstall*) de la distribución y cualquier otro material adicional que fuera necesario.

4.4. Configuración inicial de Naxtaro Linux

Una vez preparado el entorno de trabajo inicial, procedemos a personalizar la distribución, para ello elaboramos una lista de aplicaciones que queremos incluir en nuestra distribución, y que serán automáticamente instalados al hacer la construcción. Esta lista estará ubicada en un fichero que llamaremos

`naxtaro.list.chroot` dentro del directorio `~/naxtaro/config/package-lists`, con la siguiente selección de utilidades:

```
rsync
finger
telnet
joe
screen
ntpdate
bzip2
zip
debconf-utils
mlocate
dnsutils
iptables
man-db
syslog-ng
sudo
smbclient
```

A nivel de ficheros que queremos alterar o incluir, el directorio donde trabajar será el llamado `~/naxtaro/config/includes.chroot`, creamos el directorio “etc” donde incluiremos los ficheros de definición y configuración de los diferentes componentes, empezamos personalizando el nombre de la distribución creando el fichero `issue` dentro del directorio `~/naxtaro/config/includes.chroot/etc` con el contenido:

```
Naxtaro GNU/Linux v 1.0 \n \l
```

Más adelante, a medida que vayamos incluyendo más utilidades y componentes a la distribución podremos comprobar como vamos añadiendo más ficheros en este directorio.

Otro directorio importante que usaremos será el llamado *hooks,* donde incluiremos todos los comandos en forma de *shell script* que queremos que ejecute el proceso de creación de la distribución. Por ejemplo si queremos cambiar algún tipo de configuración por defecto, asignar un password en concreto a un software que vamos a instalar, cambiar algún permiso, etc lo definiremos aquí. El formato de los nombres es fundamental, han de incluir como prefijo un número que será el orden con el que se ejecutará, es decir, un número de secuencia, empezaremos por el 0500 por ejemplo, dejaremos el 0600 para el último, aquí tendremos en cuenta el orden de ejecución de los scripts, muy importante, observamos que han de acabar con la extensión llamada `.sh.chroot` para poder ser procesados.

En el apartado 5. veremos como para poder realizar la instalación automática de los distintos componentes, necesitamos definir y crear esta serie de *shellscripts.*

Para finalizar, copiaremos un logo propio y creamos los links necesarios:

```
root@naxtaro-base:~# cd ~/naxtaro/config/includes.installer/usr/share/
graphics/
root@naxtaro-base:~/naxtaro/config/includes.installer/usr/share/
graphics# ln -s logo_debian.png logo_debian_dark.png
root@naxtaro-base:~/naxtaro/config/includes.installer/usr/share/
graphics# ln -s logo_debian.png logo_installer.png
root@naxtaro-base:~/naxtaro/config/includes.installer/usr/share/
graphics# ln -s logo_debian.png logo_installer_dark.png
root@naxtaro-base:~# mkdir -p ~/naxtaro/config/includes.installer/usr/
share/graphics/
root@naxtaro-base:~# cp ~/naxtaro_linux_logo.png  ~/naxtaro/config/
includes.installer/usr/share/graphics/logo_debian.png
```

4.5. Instalación automatizada de Naxtaro Linux

El objetivo es automatizar al máximo la instalación, para ello usaremos la utilidad de Debian “preseed”[9] (preconfiguración en inglés) que ofrece un mecanismo para responder a las preguntas formuladas durante el proceso de instalación, sin necesidad que introducir manualmente las respuestas mientras la instalación está en funcionamiento. Esto hace que sea posible automatizar totalmente la mayoría de las instalaciones e incluso ofrece algunas características que no están disponibles durante una instalación normal.

Para lograr el objetivo necesitamos crear este fichero de respuestas automáticas que necesitaremos ubicar dentro del entorno de trabajo de live-build en el directorio `config/includes.installer` y con nombre `preseed.cfg`

```
### Localization
#d-i debian-installer/locale string en_US
#d-i keyboard-configuration/xkb-keymap select us

### Network configuration
d-i netcfg/enable boolean false
d-i netcfg/get_hostname string naxtaro
d-i netcfg/get_domain string naxtaro.com
d-i netcfg/hostname string naxtaro

### Mirror settings
```

```
d-i mirror/country string manual
d-i mirror/http/hostname string http.es.debian.org
d-i mirror/http/directory string /debian
d-i mirror/http/proxy string
d-i mirror/suite string stable

### Account setup
d-i passwd/root-login boolean false
d-i passwd/user-fullname string Naxtaro User
d-i passwd/username string naxtaro
d-i passwd/user-password password naxtaro10
d-i passwd/user-password-again password naxtaro10

### Clock and time zone setup
d-i clock-setup/utc boolean true
d-i time/zone string CEST
d-i clock-setup/ntp boolean true
d-i clock-setup/ntp-server string pool.ntp.org

### Partitioning
d-i partman-auto/method string regular
d-i partman-lvm/device_remove_lvm boolean true
d-i partman-md/device_remove_md boolean true
d-i partman-lvm/confirm boolean true
d-i partman-auto/choose_recipe select atomic
d-i partman/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true

### Base system installation
d-i apt-setup/non-free boolean true
d-i apt-setup/contrib boolean true
d-i apt-setup/use_mirror boolean false
d-i apt-setup/services-select multiselect security, updates
d-i apt-setup/security_host string security.debian.org

### Boot loader installation
d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean true
d-i grub-installer/bootdev string /dev/sda
d-i grub-installer/bootdev  string default

### Finishing up the installation
d-i finish-install/reboot_in_progress note
```

El fichero de configuración esta dividido en varias secciones que corresponden en orden a las secciones que el proceso de instalación va preguntando, la primera parte corresponde a la localización, lenguaje y tipo de teclado, como se observa está comentado ya que esta pregunta si que deseo que se realice, de hecho será lo único que pregunte el proceso de instalación.

La siguiente sección correspondería a las preguntas relacionadas con los datos de red: dirección IP, máscara de red, puerta de enlace, nombre del dominio etc, aquí he definido que no pregunte ningún dato ya que como veremos más adelante este proceso de configuración lo realizará el *shellscript* `naxtaro_setup.sh`. Continuaremos con la parte de configurar el repositorio más cercano a nuestra zona física, he escogido el ftp.es.debian.org, aunque esta servidor también puede ser cambiado más adelante, cuando el sistema se encuentre ya instalado. Proseguimos con la parte que preguntará sobre la creación y passwords de los usuarios, en el caso de Naxtaro no asignaremos contraseña al usuario root y solo crearemos un usuario del sistema llamado “naxtaro” con contraseña “naxtaro10”, este usuario podrá elevar los privilegios al superusuario *root* mediante el comando *sudo*. A continuación vendrá la parte donde configuramos la zona horaria y que servidor para sincronizar el reloj del sistema querremos usar, he considerado útil automatizar esta parte aunque también es configurable una vez con el sistema ya instalado. Después entramos en la parte donde nos preguntaría como particionar los discos, para simplificar el proceso definimos que use una sola partición en todo el disco disponible y que borre si hubiesen anteriores particiones de tipo *lvm* o *md*. A continuación la sección sobre que sistema base deseamos, en nuestro caso que se incluyan paquetes de software del tipo “non-free” y “contrib” y que incluya las actualizaciones disponibles de seguridad del servidor security.debian.org. El siguiente apartado hace referencia a lo que sería la instalación del llamado *boot loader*, es decir el gestor de arranque del nuevo sistema instalado, si detectase que existe otro sistema ya instalado lo añadiría a la lista de sistemas disponibles. La última sección realiza el reinicio automático cuando finaliza por completo la instalación sin ningún dialogo de confirmación.

5. Instalación y configuración de servicios

A continuación se procederá a definir y configurar los distintos componentes que he considerado como esenciales para esta primera versión de la distribución Naxtaro Linux.

5.1. Servicio de SSH

Para realizar una prueba de concepto de la instalación de servicios dentro de la distribución que queremos construir, he escogido para empezar la instalación del servicio básico de SSH. El servicio SSH nos servirá para poder conectar de forma segura y cifrada al servidor. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de

conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

Para la instalación del servicio de SSH crearemos un script dentro del directorio `hooks` llamado `0500-ssh-server.sh.chroot`. El contenido del *shellscript* será el siguiente:

```
#!/bin/bash
# instalacion del servidor ssh
apt-get -y install openssh-server
# activamos la opcion de poder hacer ssh como usuario root
echo "Disabling root login in ssh"
sed -i 's/PermitRootLogin without-password/PermitRootLogin yes/' \
/etc/ssh/sshd_config
exit 0
```

Invocamos a la utilidad de instalación de paquetes de Debian llamada *apt-get* con el parámetro "-y install" que asume afirmativamente de forma automática a las preguntas que la instalación de la aplicación pudiese hacer. A continuación, en el fichero de configuración del servidor ssh, en `/etc/ssh/sshd_config` buscamos la cadena "PermitRootLogin" y cambiamos el parámetro de "without-password" por "yes" mediante la anteriormente comentada utilidad "sed".

5.2. Servicio de Firewall

Como vimos en el apartado 4.2.3. toda configuración o scripts que pretendamos personalizar se tendrá que incluir en el directorio especial `config/includes.chroot/etc`, para que el servidor actúe como *firewall* usaremos la utilidad *iptables*[10] que precisamente se ha incluido en la lista de software que instalará el proceso de construcción. *Iptables* es una utilidad que utiliza cadenas (listas de reglas) para aceptar o denegar el tráfico. Cuando se establece una conexión al servidor, *iptables* busca en esta lista de reglas si permite el tráfico o no. Las cadenas principales son:

- INPUT: todo el tráfico entrante al servidor, pasa por esta cadena.
- OUTPUT: todo el tráfico saliente desde el servidor, pasa por esta cadena.
- FORWARD: todo el trafico enrutado a través del servidor, pasa por esta cadena.

El formato básico es

```
iptables -A REGLA -p tipo_protocolo –dport puerto -j ACCION
```

donde REGLA será INPUT, OUTPUT O FORWARD, tipo_protocolo será del tipo udp o tcp, puerto el número de puerto y ACCION si aceptamos o rechazamos el tráfico a ese puerto.

Para definir estas reglas del *firewall* crearemos un fichero tipo *shellscript* llamado `fw` (fichero que engloba en un solo fichero una serie de ordenes que podríamos ejecutar desde la linea de comandos, definiendo variables, funciones, etc). Como interesa que se ejecute cada vez que el servidor se inicie, lo colocaremos en el directorio especial llamado `init.d/` que es donde se ejecutan los comandos o scripts de inicio en un sistema Unix. Por lo tanto dentro del entorno de trabajo, creamos este fichero `fw` dentro de `~/naxtaro/config/includes.chroot/etc`, con el contenido:

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          fw
# Required-Start:    $remote_fs $syslog $network
# Required-Stop:     $remote_fs $syslog $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start iptables fw
# Description:       Enable iptables fw rules.
### END INIT INFO
# definir ubicacion del comando iptables
IPT="/sbin/iptables"

case "$1" in
start)
echo "Starting Firewalling Services: "

# activar opcion de routing
/bin/echo "1" > /proc/sys/net/ipv4/ip_forward
for each in /proc/sys/net/ipv4/conf/*
do
    echo 0 > $each/accept_redirects
    echo 0 > $each/send_redirects
done
# reducimos algunos tiempos para lograr mayor rendimiento
# y aumentamos la tabla de conexiones posibles
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 1800 > /proc/sys/net/ipv4/tcp_keepalive_time
echo 0 > /proc/sys/net/ipv4/tcp_window_scaling
echo 0 > /proc/sys/net/ipv4/tcp_sack
echo 1048576 > /proc/sys/net/ipv4/netfilter/ip_conntrack_max
echo 1048576 > /sys/module/nf_conntrack/parameters/hashsize
# vaciar y restablecer la configuración, por defecto, de iptables
$IPT -F
$IPT -t nat -F
$IPT -X
$IPT -t nat -X
# definir la politica por defecto, todo aceptado menos el trafico de entrada
$IPT -P INPUT DROP
$IPT -P FORWARD ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -t nat -P PREROUTING ACCEPT
$IPT -t nat -P POSTROUTING ACCEPT
$IPT -t nat -P OUTPUT ACCEPT
# activamos los mensajes de log
echo "Enabling logs..."
$IPT -N INPUT_DROP
```

```
$IPT -A INPUT_DROP -j LOG --log-prefix="INPUT... "
$IPT -A INPUT_DROP -j DROP
$IPT -N FORWARD_DROP
$IPT -A FORWARD_DROP -j LOG --log-prefix="FORWARD... "
$IPT -A FORWARD_DROP -j DROP
# permitir todo el trafico por la interface local
echo "Allowing localhost..."
LO="lo"
$IPT -A INPUT -i $LO -j ACCEPT
$IPT -A OUTPUT -o $LO -j ACCEPT
# interface eth0 y eth1, averiguamos las ips, se verifica que la interface
eth0 y eth1 existan
/sbin/ifconfig -a |grep eth0 &> /dev/null
# 0 si es correcto el comando, 1 si no lo es
if [ "$?" = "0" ]; then
        INTERNALIP=`/sbin/ifconfig eth0 | grep "inet addr" | cut -d B -f 1 |
cut -d : -f 2`
 INTERNALNET=`ip route | grep $INTERNALIP | awk '{print $1}'`
        echo "Configuring internal network..."
        # aceptamos trafico de la red interna
        $IPT -A OUTPUT -m state --state NEW,ESTABLISHED -j ACCEPT
        $IPT -A INPUT -m state --state NEW,ESTABLISHED -j ACCEPT
fi
/sbin/ifconfig -a |grep eth1 &> /dev/null
if [ "$?" = "0" ]; then
        EXTERNALIP=`/sbin/ifconfig eth1 | grep "inet addr" | cut -d B -f 1 |
cut -d : -f 2`
        # definimos lans de posibles sucursales
        SUCURSAL1="10.15.12.0/24"
        SUCURSAL2="10.15.13.0/24"
        echo "Configuring external network..."
        # permitimos conectar por ssh hacia el exterior
        $IPT -A OUTPUT -i eth1 -p tcp --dport 22 -m state --state
NEW,ESTABLISHED -j ACCEPT
        $IPT -A INPUT -i eth1 -p tcp --sport 22 -m state --state ESTABLISHED -
j ACCEPT
        # permitimos conectar por http/https hacia el exterior
        $IPT -A OUTPUT -i eth1 -p tcp -m multiport --dports 80,443 -m state --
state NEW,ESTABLISHED -j ACCEPT
        $IPT -A INPUT -i eth1 -p tcp -m multiports --sport 80,443 -m state --
state ESTABLISHED -j ACCEPT
        # permitimos acceso a las dns i ntp (trafico tipo udp)
        $IPT -A OUTPUT -i eth1 -p udp -m multiports --dports 53,123 -j ACCEPT
        $IPT -A INPUT -i eth1 -p udp -m multiports --sport 53,123 -j ACCEPT
        # permitimos acceso a varios servicios comunes, smtp, ntp,
        $IPT -A OUTPUT -i eth1 -p tcp -m multiports --dports 25,443 -m state
--state NEW,ESTABLISHED -j ACCEPT
        $IPT -A INPUT -i eth1 -p tcp -m multiports --sport 25,443 -m state --
state ESTABLISHED -j ACCEPT
        # para ipsec necesitaremos los puertos 500,1701 y 4500
        $IPT -A INPUT -p tcp --dport 4500 -j ACCEPT
        $IPT -A INPUT -p tcp --dport 4500 -j ACCEPT
        $IPT -A INPUT -p udp --dport 1701 -j ACCEPT
        $IPT -A INPUT -p udp --dport 4500 -j ACCEPT
        $IPT -A INPUT -p udp --dport 500 -j ACCEPT
        $IPT -A INPUT -p esp -j ACCEPT
        # definimos las internal nets que tengamos
```

```
        # sucursal1
        $IPT -t nat -A POSTROUTING -s $INTERNALNET -d $SUCURSAL2 -j ACCEPT
        # sucursal2
        $IPT -t nat -A POSTROUTING -s $INTERNALNET -d $SUCURSAL3 -j ACCEPT
        # sucursal local, el trafico por defecto lo enmascara por la ip
        # externa
        $IPT -t nat -A POSTROUTING -s $INTERNALNET -j MASQUERADE
fi
echo "done."
exit 0
;;
stop)
echo -n "Stopping Firewalling Services: "
# vaciar y restablecer la configuración, por defecto, de iptables
$IPT -t nat -F
$IPT -F
$IPT -X
# reset de la politica por defecto (aceptar todo)
$IPT -P INPUT ACCEPT
$IPT -P OUTPUT ACCEPT
$IPT -P FORWARD ACCEPT
echo "done."
exit 0
;;
status)
echo "Displaying rules..."
$IPT -L -n -v
;;
restart|reload)
echo "Restarting firewall..."
$0 stop
$0 start
echo "done."
;;
*)
echo "Usage: firewall {start|stop|status|restart|reload}"
exit 1
esac
```

Considero *eth0* como el interface correspondiente a la red interna y *eth1* para la red externa, compruebo si existen y si así aplico una serie de reglas de filtrado. Para que sea un fichero tipo `init.d` correcto, necesitará las especificaciones del principio en forma de comentarios, básicamente se define una breve descripción de lo que hace el script, en que nivel de arranque y de parada debe ser ejecutado, y de que requerimientos necesita. A continuación se define una variable que sera el path completo del comando en si. Luego mediante la cláusula *case*, se evalúa el parámetro con el que se llama al script:

- *start*: empieza el script, ejecuta una serie de opciones de configuración y las reglas básicas del trafico que está permitido. Se comprueba si existen una o dos interfaces de red, lo normal al tratarse de un servidor tipo firewall es que existan dos interfaces de red, la red externa, que provee acceso a internet y todos sus servicios, y la red

interna, que es donde ese conectaran los diferentes dispositivos de la empresa. Para que el script de firewall sea funcional en modo live o en entornos con un sólo interfaz de red he añadido la comprobación de si existe o no esta interfaz externa, pero siendo conscientes de que un firewall no tiene mucha funcionalidad con solamente una tarjeta de red instalada.

- stop: para el script, vacia y restablece la configuración, por defecto, de *iptables*, y acepta todo el tráfico (situación inicial).
- status: muestra un resumen del estado de las reglas
- restart: realiza un stop y un start del script
- *) si no se especifica ninguna opción de las anteriores muestra un pequeño mensaje de ayuda con las posibles opciones.

Como ejemplo definimos dos redes privadas de dos posibles sucursales externas, esto nos servirá para establecer conexiones *site2site* con otras sedes y que virtualmente se vean todas las redes como una única interna.
De puertos accesibles hacia el exterior dejamos el 80, 443 (http, https), el 53 para las *dns*, el 123 para en *ntp*, el 25 para poder enviar mail y el 443 que correspondería al servicio *imap*. Finalmente tenemos que habilitar una serie de puertos especiales, el 500, 1701 y 4500 para que funcione el servicio de *vpn* desde el exterior.

5.3. Servicio de apt-cacher

Como ya reflejamos en el apartado 4.2.3, un servidor proxy que guarde una copia en local de los diferentes paquetes de la distribución es sumamente útil, ahorra tiempo y ancho de banda, por lo que también lo vamos a incluir en Naxtaro. Crearemos de nuevo un script llamado `0501-apt-cacher.sh.chroot`. El contenido del *shellscript* será como el que sigue:

```
#!/bin/bash
apt-get -y install apt-cacher-ng
# seleccion de repositorio
echo "http://ftp.es.debian.org/debian/" > /etc/apt-cacher-ng/backends_debian
echo "http://security.debian.org/" >> /etc/apt-cacher-ng/backends_debian
exit 0
```

Como anteriormente, contestará “yes” automáticamente a las posibles preguntas de la instalación y creará el fichero de configuración con el servidor *mirror* de Debian más cercano, en este caso ftp.es.debian.org.

5.4. Servicio de MySQL

He escogido MySQL[11] al tratarse de un sistema de gestión de base de datos relacional, multi-usuario que utiliza el lenguaje SQL. Es un sistema muy

extendido y conocido gracias a que se trata de un sistema rápido y estable. Procedemos a la instalación de la popular base de datos.

Crearemos un script dentro de este directorio llamado `0502-mysql-server.sh.chroot`. El contenido del *shellscript* será el siguiente:

```
#!/bin/bash
echo "mysql-server-5.5 mysql-server/root_password password
naxtaro2016" | debconf-set-selections
echo "mysql-server-5.5 mysql-server/root_password_again password
naxtaro2016" | debconf-set-selections
apt-get -y install mysql-server-5.5
exit 0
```

Como se puede leer en el script, se instala el aplicativo MySQL-server versión 5.5 y asignamos el password al usuario root o administrador “naxtaro2016”. En el apartado 4.2.3 en la lista de “package-list” incluimos para instalar la aplicación “debconf-utils”, que incluye “defconf-set-selections” que sirve para poder responder preguntas durante el proceso de instalación de manera automática. Para lograr el objetivo de la creación de la distribución es fundamental que ésta se produzca de la manera mas automática posible, sin tener que ir preguntando lo mismo cada vez. Por ello es muy importante automatizar todo el proceso lo máximo posible.

5.5. Servicio de DHCP

El protocolo de configuración dinámica de host (DHCP[12], *Dynamic Host Configuration Protocol*) es un estándar TCP/IP diseñado para simplificar la administración de la configuración IP de los equipos de nuestra red.

Si disponemos de un servidor DHCP, la configuración IP de los dispositivos de nuestra red puede hacerse de forma automática, evitando así la necesidad de tener que realizar manualmente uno por uno la configuración TCP/IP de cada equipo.

Un servidor DHCP es un servidor que recibe peticiones de clientes solicitando una configuración de red IP. El servidor responderá a dichas peticiones proporcionando los parámetros que permitan a los clientes autoconfigurarse. Para que un cliente solicite la configuración a un servidor, en la configuración de red de los dispositivos hay que seleccionar la opción “Obtener dirección IP automáticamente”. El servidor proporcionará al cliente al menos una dirección IP y su correspondiente mascara de red, también podrá enviar otros parámetros como la puerta de enlace, los servidores *dns* que se usarán, etc.

El servidor DHCP utiliza un modelo cliente-servidor en el que mantiene una administración centralizada de las direcciones IP utilizadas en la red. Los

clientes podrán solicitar al servidor una dirección IP y así poder integrarse en la red. El servidor DHCP proporciona una configuración de red TCP/IP segura y evita conflictos de direcciones repetidas. Solamente asigna direcciones dentro de un rango definido. Si por error hemos configurado manualmente una IP estática perteneciente al rango gestionado por nuestro servidor DHCP, podría ocurrir que dicha dirección sea asignada dinámicamente a otro dispositivo, provocándose un conflicto de IP. En ese caso el cliente solicitará y comprobará, otra dirección IP, hasta que obtenga una dirección IP correcta que no esté asignada actualmente a ningún otro equipo de la red.

La primera vez que arranquemos un dispositivo con configuración DHCP, pasará a convertirse en un cliente DHCP e intentará localizar un servidor DHCP para obtener una configuración desde él mismo. Si no encuentra ningún servidor DHCP, el cliente no podrá disponer de dirección IP y por lo tanto no podrá comunicarse con la red. Si el cliente encuentra un servidor DHCP, éste le proporcionará, para un periodo predeterminado, una configuración IP que le permitirá comunicarse con la red. Básicamente le asignara una determinada dirección durante un determinado rango de tiempo. Toda estas opciones las definiremos en nuestro servidor DHCP.

De nuevo en el directorio llamado *hooks* incluiremos todos los comandos en forma de *shellscript* que queremos que ejecute el proceso de creación de la distribución. Siguiendo el mismo procedimiento que en el apartado anterior el nombre será `0503-dhcpd-server.sh.chroot`

```
#!/bin/bash
apt-get -y install isc-dhcp-server
exit 0
```

En el directorio `config/includes.chroot/etc/dhcp` pondremos la configuración personalizada siguiente, observamos como se define una serie de variables dentro del fichero de configuración, estas variables se substituirán mediante el script de instalación `naxtaro_setup.sh`

```
option domain-name "_INTERNALDOMAIN_";
option domain-name-servers _INTERNALIP_;

# tiempos de asignacion de las IPs
default-lease-time 600;
max-lease-time 7200;

# definicion del rango de red interna
subnet _INTERNALSUBNET_ netmask _INTERNALMASK_ {
  range _INTRANGE_START_ _INTRANGE_END_;
  option routers _INTERNALIP_;
  option subnet-mask              _INTERNALMASK_;
  option domain-name              "_INTERNALDOMAIN_";
  option domain-search      "_INTERNALDOMAIN_","_EXTERNALDOMAIN_";
}
```

```
# si interesa asignar una ip especifica a un elemento determinado, por
# ejemplo una impresora, podriamos hacerlo especificando su direccion
# arp:
#host printer1.nax.es {
#         hardware ethernet a0:99:92:1a:7a:bc;
#         fixed-address 10.15.13.3;
#}
```

5.6. Servicio de DNS

DNS[13][14] es una abreviatura que significa sistema de nombres de dominio (*Domain Name System*, en inglés), se trata de un sistema para asignar nombres a equipos y servicios de red que se organiza en una jerarquía de dominios. La asignación de nombres DNS se utiliza en las redes TCP/IP, como Internet, para localizar equipos y servicios con nombres descriptivos. Cuando un usuario escriba una dirección de Internet en una aplicación, los servicios DNS podrán traducir el nombre a otra información asociada con el mismo, como una dirección IP.

Cuando el servidor DNS recibe una consulta, primero comprueba si puede responder la consulta con autoridad en función de la información de registro de recursos contenida en una zona configurada localmente en el servidor. Si el nombre consultado coincide con un registro de recursos correspondiente en la información de zona local, el servidor responde con autoridad y usa esta información para resolver el nombre consultado.

Si no existe ninguna información de zona para el nombre consultado, el servidor comprueba si puede resolver el nombre mediante la información almacenada en la caché local de consultas anteriores. Si aquí se encuentra una coincidencia, el servidor responde con esta información. De nuevo, si el servidor preferido puede responder al cliente solicitante con una respuesta coincidente de su caché, finaliza la consulta.

Los nombres de dominio completamente calificados o FQDN (*Fully Qualified Domain Name*) se componen por lo general del nombre del host, un nombre de dominio secundario y un nombre de dominio primario o de nivel máximo (*top-level domain*), que son secciones organizadas jerárquicamente

La estructura básica del DNS es similar a un árbol, donde se tiene una raíz o root, los dominios de nivel principal (*Top Level Domains*) y los dominios de segundo nivel.

Existen varios productos que realizan esta función y en todas las plataformas, pero el más usado es BIND (Berkeley Internet Name Domain), que es distribuido bajo la GNU GPL y es que usaré para Naxtaro Linux.

Como en la instalación de los servicios anteriores, se crea un script llamado `0504-bind9-server.sh.chroot` con el contenido,

```
#!/bin/bash
apt-get -y install bind9
exit 0
```

En el directorio `config/includes.chroot/etc/bind` pondremos los correspondientes ficheros de configuración, el general llamado `named.conf.options`,

```
options {
        directory "/var/cache/bind";
        // servidores dns de nuestro proveedor o los de google
        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
        dnssec-validation auto;
        auth-nxdomain no;    // conform to RFC1035
        // allow-query no lo restringuiremos, ya que se trata de un
        // servidor dns publico que gestionara nuestro dominio
        // allow-recursion: solo la autorizamos a la lista de ip's
        // permitidas, normalmente la red interna
        // allow-transfer: autorizaremos la ip del dns secundario, en
        // caso de disponer de el
        // allow-notify:  autorizaremos la ip del dns secundario, en
        // caso de disponer de el
        allow-query {};
        allow-recursion { 127.0.0.1; _INTERNALNET_; };
        allow-transfer { 2.2.2.2; };
        allow-notify { 2.2.2.2; };
        // definimos parametros y opciones de seguridad
        dnssec-enable yes;
        statistics-file "named.stats";
        zone-statistics yes;
        version         "[naxtaro-dns]";
        notify yes;
        transfer-format many-answers;
        max-transfer-time-in 60;
        interface-interval 0;
};
//opciones de logging en ficheros separados
logging {
        channel queries_channel {
                file "/var/log/named/queries" versions 5 size 30m;
                print-time yes;          // timestamp log entries
        };
        channel zone-xfer_channel {
                file "/var/log/named/zone-xfer" versions 5 size 30m;
                print-time yes;
        };
        channel default_channel {
                file "/var/log/named/named.log" versions 5 size 30m;
```

```
 print-time yes;
                severity info;
        };
        category default {
                default_channel;
        };
        category queries {queries_channel; };
        category xfer-out {zone-xfer_channel; };
        category xfer-in {zone-xfer_channel; };
        category notify {zone-xfer_channel; };
};
    category notify {zone-xfer_channel; };
};
```

En la primera parte configuramos los servidores DNS de nuestro proveedor de servicios de Internet, o alguno público que funcionen correctamente como los de Google, luego se definen varios parámetros de seguridad, básicamente se restringe el acceso al servicio a la red interna y se define un servidor secundario DNS con ip de ejemplo 2.2.2.2, que estaría autorizado a recibir las zonas del primario. La última parte se define para obtener más detalle de la actividad del servicio, organizada en varios ficheros de log.

El siguiente fichero que necesitaremos se llamará named.conf.local, que definirá un ejemplo de dominio público que hemos registrado y podemos configurar llamado “naxtaro.com”, y otro privado solamente accesible desde la red interna llamado “naxtaroint.com”.

```
zone "naxtaro.com" {
        type master;
        file "/etc/bind/naxtaro.com";
        allow-query { any; };
};

zone "naxtaroint.com" {
        type master;
        file "/etc/bind/naxtaroint.com";
        allow-query { _INTERNALNET_; };
};
```

Se crean los dos ficheros `naxtaro.com` y `naxtaroint.com` que corresponderían a estos dos dominios, con el contenido

```
$ORIGIN naxtaro.com.
$TTL 900 ; 15 mins tiempo de expiracion
@       IN      SOA     ns1.naxtaro.com.   hostmaster.naxtaro.com. (
                                2016021866      ; serial
                                43200           ; refresh
                                1800            ; retry
                                604800          ; expire
```

```
                                300             ; minimum
                                )

        IN      NS      ns1.naxtaro.com.   ; servidor dns primario
        IN      NS      ns2.naxtaro.com.   ; servidor dns secundario
        IN      MX      10 mail.naxtaro.com  ; servidor de mail
ns1     IN      A       _EXTERNALIP_   ; ip del servidor dns primario
ns2     IN      A       2.2.2.2   ; ip del servidor dns secundario
mail    IN      A       3.3.3.3   ; ip del servidor de mail
www     IN      A       4.4.4.4   ; ip del servidor web
```

```
$ORIGIN naxtaroint.com.
$TTL 900 ; 15 mins tiempo de expiracion
@       IN      SOA     ns1.naxtaroint.com.
hostmaster.naxtaroint.com. (
                                2016021866      ; serial
                                43200           ; refresh
                                1800            ; retry
                                604800          ; expire
                                300             ; minimum
                                )

        IN      NS      ns1.naxtaroint.com.   ; servidor dns primario
ns1     IN      A       _INTERNALIP_   ; ip del servidor dns primario
www     IN      A       5.5.5.5   ; ip del servidor web
git     IN      A       6.6.6.6   ; ip servidor git
```

El formato del fichero es básicamente una tabla que define el nombre de nuestros hosts y su correspondiente dirección IP. El TTL es el tiempo de refresco, es decir definimos cuanto tiempo pueden cachear como máximo las entradas de esta zona otros DNS, a partir de 900 segundos o 15 minutos deberían consultar de nuevo las definiciones. A continuación tenemos varios valores numéricos que equivalen a segundos como un número de serie que debemos incrementar cada vez que editamos o añadimos alguna entrada, tiempo de refresco, tiempo de reintento, tiempo de expiración y tiempo mínimo. A modo de ejemplo se definen unos cuantos hosts como un supuesto DNS secundario llamado *ns2* y que resolvería con la IP 2.2.2.2, un servidor de mail con la 3.3.3.3 o un posible servidor web que con el nombre de www.naxtaro.com resolvería con la IP 4.4.4.4. La IP del servidor principal, que tendría el nombre de ns1.naxtaro.com definida mediante la variable _EXTERNALIP_ la obtendremos al instalar la distribución y ejecutar el *shellscript* `naxtaro_setup.sh`. En el dominio interno naxtaroint.com se definen algunos nombres de hosts internos a modo de ejemplo web o git y la variable _INTERNALIP_ obviamente la configurará el mismo *shellscript*.

5.7. Servicio de VPN y *site2site*

Un servicio de VPN[15][16][17] (*Virtual Private Network*) permite crear una conexión segura a otra red a través del Internet. Cuando se conecta un

dispositivo a una VPN, este actúa como si estuviese en la misma red que la que tiene el VPN y todo el tráfico de datos se envía de forma segura a través del VPN. En nuestro caso nos será de utilidad para poder utilizar los recursos de la empresa como si estuviésemos fisicamente en ella.

A través de esta conexión VPN pasará información privada y confidencial que en las manos equivocadas, podría resultar perjudicial para cualquier empresa. Esto se agrava aún más si el empleado en cuestión se conecta utilizando una red Wi-Fi pública abierta. Afortunadamente, este problema puede ser mitigado cifrando los datos que se envían y reciben. Para poder lograr este objetivo, se pueden utilizar los siguientes protocolos:

- **IPsec** (Internet Protocol Security): permite mejorar la seguridad a través de algoritmos de cifrado robustos y un sistema de autentificación más exhaustivo. IPsec posee dos métodos de encriptado, modo transporte y modo túnel. Asimismo, soporta encriptado de 56 bit y 168 bit (triple DES).
- **PPTP/MPPE**: tecnología desarrollada por un consorcio formado por varias empresas. PPTP soporta varios protocolos VPN con cifrado de 40 bit y 128 bit utilizando el protocolo Microsoft Point to Point Encryption (MPPE). PPTP por sí solo no cifra la información.
- **L2TP/IPsec** (L2TP sobre IPsec): es capaz de proveer el nivel de protección de IPsec sobre el protocolo de túnel L2TP. Al igual que PPTP, L2TP no cifra la información por sí mismo.

Parte de la protección de la información que viaja por una VPN es el cifrado, no obstante, verificar que la misma se mantenga íntegra es igual de importante. Para lograr esto, IPsec emplea un mecanismo que si detecta alguna modificación dentro de un paquete, procede a descartarlo. Proteger la confidencialidad e integridad de la información utilizando una VPN es una buena medida para navegar en Wi-Fi públicos e inseguros incluso si no se desea acceder a un recurso corporativo.

Muchos sistemas operativos modernos llevan incorporado el soporte para poder configurar una conexión L2TP / IPSec VPN. Esto permite la creación de una VPN a través de Android , Windows , Linux , MacOS y otros sistemas operativos sin ningún tipo de requisitos de software comercial ni tener que instalar ningún tipo de software adicional.

Por lo tanto será L2TP sobre IPsec la solución que implementaré para Naxtaro Linux, a través del software llamado “strongSwan”, que es básicamente un *daemon* de claves, que utiliza los protocolos de Internet Key Exchange (IKEv1 y IKEv2) para establecer asociaciones de seguridad (SA) entre dos clientes.

Básicamente serán tres los componentes necesarios para poder implementarlo:

- strongSwan que nos proveerá de la seguridad en la comunicación entre el cliente y la red,

- xl2tpd que nos crea un túnel en l que el trafico VPN fluye entre IPsec de una manera transparente,
- ppp (Point-to-Point Protocol) que se encargará de la autenticación de los usuarios que conectan a la VPN

Estos tres componentes son los que veremos reflejados en el correspondiente *shellscript*, llamado `0505-strongswan.sh.chroot`,

```
#!/bin/bash
apt-get -y install strongswan
apt-get -y install xl2tpd
apt-get -y install ppp
exit 0
```

La configuración de *strongswan* va ubicada directamente en el directorio *etc*, por lo tanto en `config/includes.chroot/etc` pondremos el fichero `ipsec.conf`

A grandes rasgos se definen dos tipos de conexiones, la correspondiente a clientes que se van a conectar a la VPN con o sin NAT y dos conexiones de ejemplo que serían las que conectarían a otras sucursales, lo que es conocido por *site2site.* Para *strongswan* el lado *left* seria nuestra red y el *right* al que queremos conectar. Como se intuye la configuración en las otras sucursales sería idéntica pero intercambiando los datos de *left* y *right*.

A continuación, debemos crear el fichero `ipsec.secrets` donde vendrán las claves públicas que se usarán para autenticación

```
_EXTERNALIP_ _SUCURSAL1EXTERNALIP_ : PSK 'SuperStrongPassword2016'
_EXTERNALIP_ _SUCURSAL2EXTERNALIP_ : PSK 'SuperStrongPassword2016'
%any : PSK "ClientsSuperStrongPassword2016"
```

Para las conexiones entre sucursales, o el llamado *site2site*, la clave pública será “SuperStrongPassword2016” y para los dispositivos o clientes que conectarán al servicio de VPN será “ClientSuperStrongPassword2016”.

Para el servicio de xl2ptd, definiremos en el directorio `config/includes.chroot/etc/xl2tpd/xl2tpd.conf`

```
[global]
ipsec saref = yes

[lns default]
ip range = _INTRANGE2_START_-_INTRANGE2_END_
local ip = _INTERNALIP_
unix authentication = yes
require authentication = yes
ppp debug = yes
pppoptfile = /etc/ppp/options.xl2tpd
length bit = yes
name = l2tpd
```

Igual que el servicio de DHCP aquí también definimos un rango de direcciones que asignaremos a los clientes. Para autenticar a estos clientes usaremos el sistema “unix authentication” es decir consultaremos los usuarios y password locales del propio servidor que residirán en el fichero `/etc/passwd`.

```
conn L2TP-PSK-NAT
    rightsubnet=vhost:%priv
    also=L2TP-PSK-noNAT

conn L2TP-PSK-noNAT
    authby=secret
    pfs=no
    auto=add
    keyingtries=3
    rekey=no
    ikelifetime=8h
    keylife=1h
    dpddelay=30
    dpdtimeout=120
    dpdaction=clear
    type=transport
    left=_EXTERNALIP_
    leftprotoport=17/1701
    right=%any
    rightprotoport=17/%any

conn CENTRAL-SUCURSAL1
        leftfirewall=yes
        lefthostaccess=yes
        left=           _EXTERNALIP_
        leftsubnet=     _INTERNALNET_
        right=          _SUCURSAL1EXTERNALIP_
        rightsubnet=    _SUCURSAL1INTERNALNET_
        dpdaction=restart
        dpddelay=30
        dpdtimeout=120
        authby=secret
        auto=start
        fragmentation=yes

conn CENTRAL-SUCURSAL2
        leftfirewall=yes
        lefthostaccess=yes
        left=           _EXTERNALIP_
        leftsubnet=     _INTERNALNET_
        right=          _SUCURSAL2EXTERNALIP_
        rightsubnet=    _SUCURLSAL2INTERNALNET_
        dpdaction=restart
        dpddelay=30
        dpdtimeout=120
        authby=secret
```

```
        auto=start
        fragmentation=yes
```

Las opciones del *ppp* las especificamos en dos ficheros separados, `config/includes.chroot/etc/ppp/options.xl2tpd`

```
name l2tpd
ms-dns _INTERNALIP_
ms-dns 8.8.8.8
noccp
crtscts
idle 18000
mtu 1280
mru 1280d
nodefaultroute
lock
proxyarp
connect-delay 5000
auth
login
require-pap
refuse-eap
refuse-chap
refuse-mschap
refuse-mschap-v2
```

y el `config/includes.chroot/etc/ppp/pap-secrets`

```
*       l2tpd           ""              *
```

y por último, como usaremos autenticación local a partir del fichero *passwd* y *shadow* del propio Linux, incluiremos el fichero modificado `ppp` dentro de `config/includes.chroot/etc/pam.d`

```
auth    required        pam_nologin.so
auth    required        pam_unix.so
account required        pam_unix.so
session required        pam_unix.so
```

5.8. Servicio de NTP

Un servicio de NTP[18] (*Network Time Protocol*) permite sincronizar los relojes de los dispositivos que lo consulten. Es muy útil cuando tenemos un conjunto de equipos informáticos que queremos que se encuentren todos sincronizados exactamente a la misma hora. El servicio lo instalamos de la forma habitual, creando el *shellscript* `0506-ntp.sh.chroot`

```
#!/bin/bash
apt-get -y install ntp
exit 0
```

A continuación en `naxtaro_setup.sh` se ejecutará el comando `sed -i s%192.168.123.255%$INTLOCALIP_FIRST.255%g /etc/ntp.conf` es decir, dentro de la configuración `ntp.conf` substituye la red de ejemplo por la nuestra. Como en este caso substituimos directamente la única linea del fichero de configuración que nos interesa, no es necesario copiar el fichero dentro de `includes.chroot/etc`.

5.9. Servicio de Webproxy

Un servicio de *webproxy* nos permitirá mantener una copia local del contenido más frecuente que visiten los diferentes dispositivos de nuestra red local. En concreto se ha escogido el popular software de proxy "Squid"[19], que soporta los protocolos http, https, ftp entre otros. Gracias a este proxy se reducirá el ancho de banda de la red y mejorará el tiempo de respuesta de los servidores ya que estarán cacheados los contenidos en el servidor.

Como en anteriores servicios creamos el fichero con nombre `0507-squid3.sh.chroot` en el directorio *hooks*, y contenido

```
#!/bin/bash
apt-get -y install squid3
exit 0
```

Igual que el servicio de NTP, la configuración se modificará directamente cuando ejecutemos el *shellscript* `naxtaro_setup.sh` substituyendo los valores de red necesarios.

5.10. Servicio de ficheros

Gracias al software *samba*[20] podremos ofrecer un servicio de creación de unidades de red a los dispositivos de la red local, y de esta manera poder compartir ficheros, documentos etc. De forma habitual, creamos el *shellscript* llamado `0508-samba.sh.chroot` con

```
#!/bin/bash
apt-get -y install samba
exit 0
```

Este caso es un poco diferente a los anteriores, ya que se optará por substituir variables dentro del fichero de configuración directamente, sino que se substituirá directamente el fichero de configuración entero, entonces creamos el directorio `config/includes.chroot/etc/samba` y se copia directamente allí el fichero `smb.conf` añadiendo al final del mismo un par de directorios de ejemplo que serían los que compartiríamos en la red.

El primero, a nivel de servidor se compartiría el directorio llamado `/datos` y sería de acceso privado y sólo podría ser accedido por los usuarios autenticados. En el segundo se compartiría el directorio `/misc` y cualquier dispositivo de la red local podría acceder sin necesidad de autenticarse.

```
[datos]
   comment = datos
   path = /datos
   browseable = yes
   writable = yes
   public = no
   valid users = naxtaro

[misc]
   comment = miscellaneous
   path = /misc
   browseable = yes
   writable = yes
   public = yes
```

5.11. Servicio de Alta Disponibilidad

Incluiremos este último pero importante servicio. Algunas empresas necesitan una serie de servicios funcionales las 24 horas del día, 7 días a la semana, los 365 días del año. Por ejemplo hay empresas que necesitan conexión a internet permanente para poder trabajar, o necesitan tener siempre disponible una base de datos, o el servidor *dhcp* para que los dispositivos puedan seguir recibiendo direcciones IP y todo siga funcionando normalmente. Por lo tanto, no podemos confiar esta serie de servicios críticos a un sólo servidor, ya que en cualquier momento y sobretodo a medida que pase el tiempo podría fallarle la fuente de alimentación, el disco duro, algún módulo de memoria, etc. También podría ocurrir que algún servicio dejase de funcionar correctamente, el software no es perfecto!.

Para este tipo de exigencias en la disponibilidad existen varias soluciones disponibles para Linux. En el caso de Naxtaro se ha optado por el proyecto Linux-HA[21] (http://www.linux-ha.org/). Construiremos un *clúster* en configuración Activo/Pasivo en la que el primer servidor será el nodo maestro, y el segundo será el nodo de respaldo o de backup. El trabajo del nodo de respaldo es hacerse cargo de los servicios del nodo maestro si cree que dicha máquina no está respondiendo. Ambos servidores se intercambiarán esta información por el cable de red, comprobando uno al otro cada poco tiempo si está “vivo” mediante lo que llaman “latidos” (*hearthbeat*, en inglés), si el nodo de respaldo no puede contactar con el servidor maestro, pensará que éste ha dejado de funcionar y tomará su dirección IP y sus servicios asociados.

Si disponemos de presupuesto lo ideal sería instalar una tarjeta de red adicional para que se ocupase únicamente de estos “latidos” pero para simplificar, en Naxtaro usaremos la interfaz interna.

Siguiendo el proceso de los anteriores servicios, crearemos un *shellscript* en el directorio *hooks* con la instrucción de instalar el software de alta disponibilidad que se llama *heartbeat*, `0509-heartbeat.sh.chroot`

```
#!/bin/bash
apt-get -y install heartbeat
exit 0
```

Los ficheros principales de configuración que crearemos en `config/includes.chroot/etc/ha.d` serán:

- `authkeys`: es el fichero de autenticación, para seguridad la palabra clave deberá coincidir en ambos servidores, y los permisos del fichero deberán ser de lectura, escritura y ejecución restringidos al usuario root (`chmod 700`).

```
auth 3
3 md5 naxtaro2016secretpassword
```

- `ha.cf`: es el fichero de configuración principal, define el nombre de los nodos del *cluster* (naxtaro1 y naxtaro2), que puerto se usará para los “latidos”, la frecuencia de éstos, tiempos en los que se determinará que el segundo nodo no responde y se deberá tomar el control y parámetros de compresión:

```
keepalive 2
warntime 30
deadtime 60
auto_failback on
udpport 2694
bcast eth0
node naxtaro1 naxtaro2
use_logd yes
conn_logd_time 60
compression bz2
compression_threshold 2
traditional_compression false
```

- `haresources`: define de que servicios será responsable el nodo principal:

```
naxtaro1 IPaddr2::_INTERNALIP_/eth0 fw sshd apt-cacher-ng mysqld \
isc-dhcp-server bind9 ipsec ntpd squid3 samba
```

en este caso el *shellscript* `naxtaro-setup.sh` substituirá la variable _INTERNALIP_ por su correspondiente dirección IP cuando el servidor se ejecute. También se crean unos *symbolic links* dentro de `config/includes.chroot/etc/ha.d/resource.d/` que apuntarán a los scripts que arrancan o paran los distintos servicios que nos interesan que se inicien en caso de que el nodo de respaldo tenga que arrancarlos:

```
lrwxrwxrwx 1 root root 25 May  6 13:27 apt-cacher-ng -> /etc/init.d/
apt-cacher-ng
lrwxrwxrwx 1 root root 17 May  6 13:28 bind9 -> /etc/init.d/bind9
lrwxrwxrwx 1 root root 14 May  6 13:28 fw -> /etc/init.d/fw
lrwxrwxrwx 1 root root 17 May  6 13:28 ipsec -> /etc/init.d/ipsec
lrwxrwxrwx 1 root root 27 May  6 13:28 isc-dhcp-server -> /etc/init.d/
```

```
isc-dhcp-server
lrwxrwxrwx 1 root root 17 May  6 13:29 mysql -> /etc/init.d/mysql
lrwxrwxrwx 1 root root 15 May  6 13:29 ntp -> /etc/init.d/ntp
lrwxrwxrwx 1 root root 17 May  6 13:29 samba -> /etc/init.d/samba
lrwxrwxrwx 1 root root 18 May  6 13:29 squid3 -> /etc/init.d/squid3
lrwxrwxrwx 1 root root 15 May  6 13:29 ssh -> /etc/init.d/ssh
```

En resumen, *heartbeat* realizará una comprobación cada 2 segundos sobre el segundo servidor, si en 30 segundos no obtiene respuesta generará un aviso (*warning*) y determinará que el nodo principal está muerto en 60 segundos. Entonces se asignará de manera automática la dirección IP del nodo principal e iniciará los servicios definidos en el fichero `haresources` y cuyo *link* dentro de `resource.d` se haya creado.

6. Obtención de datos de red

Una parte esencial para que todo funcione y se configure correctamente es la recopilación de datos de red necesarios, para ello incluiremos en la ubicación habitual `config/includes.chroot/bin` el *shellscript* `naxtaro_setup.sh`

```
#!/bin/bash
clear
echo "Welcome to Naxtaro Linux v1.0 setup"
echo ""
echo "Please make sure all the ethernet cables are connected to the
right devices,"
echo "eth1 card to external network and eth0 attached to internal
network"
echo "============================"
echo "= External network setup ="
echo "============================"
echo -n "External IP address (10.10.10.149)? "
read EXTERNALIP
if [ "$EXTERNALIP" = "" ]; then
   EXTERNALIP="10.10.10.149"
fi
echo -n "External mask (255.255.255.0)? "
read EXTERNALMASK
if [ "$EXTERNALMASK" = "" ]; then
        EXTERNALMASK="255.255.255.0"
fi
echo -n "External default gateway (10.10.10.1)? "
read EXTERNALGW
if [ "$EXTERNALGW" = "" ]; then
        EXTERNALGW="10.10.10.1"
fi
echo -n "External domain name (naxtaro.com)? "
read EXTERNALDOMAIN
if [ "$EXTERNALDOMAIN" = "" ]; then
        EXTERNALDOMAIN="naxtaro.com"
fi

echo "============================"
echo "= Internal network setup ="
echo "============================"
echo -n "Internal IP address (10.15.13.1)? "
```

```
read INTERNALIP
if [ "$INTERNALIP" = "" ]; then
   INTERNALIP="10.15.13.1"
fi
echo -n "Internal domain name (naxtaroint.com)? "
read INTERNALDOMAIN
if [ "$INTERNALLDOMAIN" = "" ]; then
        INTERNALDOMAIN="naxtaroint.com"
fi
echo "Setting network, please wait..."
# more useful variables
EXTERNALNET=`ip route | grep $EXTERNALIP | awk '{print $1}'`
#(10.10.10.0/24)
INTERNALNET=`ip route | grep $INTERNALIP | awk '{print $1}'`
#(10.15.13.0/24)
EXTERNALSUBNET=`ip route |grep $EXTERNALNET | awk '{print $1}' | head
-1 | cut -f1 -d/` #(10.10.10.0)
INTERNALSUBNET=`ip route |grep $INTERNALNET | awk '{print $1}' | head
-1 | cut -f1 -d/` #(10.15.13.0)
INTLOCALIP_LAST=`echo $INTERNALIP|cut -f4 -d.` #(1)
INTLOCALIP_FIRST=`echo $INTERNALIP|cut -f1,2,3 -d.` #(10.15.13)
INTRANGE_START="$INTLOCALIP_FIRST.$(($INTLOCALIP_LAST+9))"
#(10.15.13.10)
INTRANGE_END="$INTLOCALIP_FIRST.$(($INTLOCALIP_LAST+199))"
#(10.15.13.200)
INTRANGE2_START="$INTLOCALIP_FIRST.$(($INTLOCALIP_LAST+200))"
#(10.15.13.201)
INTRANGE2_END="$INTLOCALIP_FIRST.254" #(10.15.13.254)

sed -i s%_EXTERNALIP_%$EXTERNALIP%g /etc/network/interfaces.naxtaro
sed -i s%_EXTERNALMASK_%$EXTERNALMASK%g /etc/network/
interfaces.naxtaro
sed -i s%_EXTERNALGW_%$EXTERNALGW%g /etc/network/interfaces.naxtaro
sed -i s%_EXTERNALDOMAIN_%$EXTERNALDOMAIN%g /etc/network/
interfaces.naxtaro
sed -i s%_INTERNALIP_%$INTERNALIP%g /etc/network/interfaces.naxtaro
sed -i s%_INTERNALMASK_%$INTERNALMASK%g /etc/network/
interfaces.naxtaro
sed -i s%_INTERNALDOMAIN_%$INTERNALDOMAIN%g /etc/network/
interfaces.naxtaro
cp -f /etc/network/interfaces.naxtaro /etc/network/interfaces

# aplying definitions
echo "domain $INTERNALDOMAIN" > /etc/resolv.conf
echo "search $INTERNALDOMAIN $EXTERNALDOMAIN" >> /etc/resolv.conf
echo "nameserver $INTERNALIP" >> /etc/resolv.conf
/etc/init.d/networking restart

# dhcpd service
sed -i s%_INTERNALDOMAIN_%$INTERNALDOMAIN%g /etc/dhcp/dhcpd.conf
sed -i s%_INTERNALIP_%$INTERNALIP%g /etc/dhcp/dhcpd.conf
sed -i s%_INTERNALSUBNET_%$INTERNALSUBNET%g /etc/dhcp/dhcpd.conf
sed -i s%_INTERNALMASK_%$INTERNALMASK%g /etc/dhcp/dhcpd.conf
sed -i s%_INTRANGE_START_%$INTRANGE_START%g /etc/dhcp/dhcpd.conf
sed -i s%_INTRANGE_END_%$INTRANGE_END%g /etc/dhcp/dhcpd.conf
/etc/init.d/isc-dhcp-server restart

# ipsec service
sed -i s%_EXTERNALIP_%$EXTERNALIP%g /etc/ipsec.conf
sed -i s%_INTERNALNET_%$INTERNALNET%g /etc/ipsec.conf
sed -i s%_EXTERNALIP_%$EXTERNALIP%g /etc/ipsec.secrets
sed -i s%_INTERNALIP_%$INTERNALIP%g /etc/ppp/options.xl2tpd
sed -i s%_INTRANGE2_START_%$INTRANGE2_START%g /etc/xl2tpd/xl2tpd.conf
```

```
sed -i s%_INTRANGE2_END_%$INTRANGE2_END%g /etc/xl2tpd/xl2tpd.conf
sed -i s%_INTERNALIP_%$INTERNALIP%g /etc/xl2tpd/xl2tpd.conf
/etc/init.d/ipsec restart

# dns service
sed -i s%_INTERNALNET_%$INTERNALNET%g /etc/bind/named.conf.options
sed -i s%_INTERNALNET_%$INTERNALNET%g /etc/bind/named.conf.local
sed -i s%_EXTERNALIP_%$EXTERNALIP%g /etc/bind/naxtaro.com
sed -i s%_INTERNALIP_%$INTERNALIP%g /etc/bind/naxtaroint.com
/etc/init.d/bind9 restart

# ntp service
sed -i s%"\#broadcast 192.168.123.255"%"broadcast $INTLOCALIP_FIRST.
255"%g /etc/ntp.conf
/etc/init.d/ntp restart

# webproxy service
sed -i s%"\#acl localnet src 10.0.0.0/8"%"acl localnet src
$INTERNALNET"%g /etc/squid3/squid.conf
sed -i s%"\#http_access allow localnet"%"http_access allow localnet"%g
/etc/squid3/squid.conf
/etc/init.d/squid3 restart
# ha service
sed -i s%_INTERNALNET_%$INTERNALNETP%g /etc/ha.d/haresources
/etc/init.d/heartbeat restart
exit 0
```

El script nos preguntará cual será la dirección IP externa, la máscara de red, su puerta de enlace (*gateway*) y que dominio usaremos. Para la red interna deberemos introducir también esta misma información. A partir de estos datos de red iniciales el script substituirá las variables correspondientes en el fichero que antes habíamos personalizado con la configuración de red del servidor, llamado `interfaces.naxtaro`. Gracias al poderoso comando *sed* y su parámetro "-i" es posible realizar el cambio directamente en el fichero. El resto del *script* irá ejecutando similares operaciones, calculando variables que necesita y aplicándolas en los ficheros de configuración según el servicio que corresponda: *dhcpd, ipsec, dns, ntp, webproxy* y *ha*. Por lo tanto dentro de `config/includes.chroot/etc/network` incluiremos el fichero `interfaces.naxtaro` con las variables necesarias,

```
# The loopback network interface
auto lo
iface lo inet loopback

# Internal network interface
allow-hotplug eth0
iface eth0 inet static
       address _EXTERNALIP_
       netmask _EXTERALMASK_
       gateway _EXTERNALGW_
       dns-nameservers _INTERNALIP_
       dns-search _EXTERNALDOMAIN_

# External network interface
allow-hotplug eth1
iface eth1 inet static
```

```
        address _INTERNALIP_
        netmask _INTERNALMASK_
        dns-nameservers _INTERNALIP_
        dns-search _INTERNALDOMAIN_
```

7. Construcción de Naxtaro Linux

Para construir el fichero *iso* de la distribución, volveremos a la utilidad *livebuild*[22], esta vez la invocamos mediante el parámetro “build”. Este proceso de construcción se divide en distintas etapas que se van ejecutando en orden. La primera etapa inicial que se ejecuta es la etapa de arranque, llamada *bootstrap*. En esta fase se construye dentro del directorio *chroot* los paquetes necesarios para la creación del sistema. La segunda fase es la fase *chroot*, que completa la fase anterior con los paquetes extra que hemos definido para personalizar la distribución. La fase final es la llamada *binary*, que construye la imagen usando los contenidos del directorio *chroot* de la fase anterior para construir el sistema de ficheros raíz, en esta fase se añade el instalador *debinstall* de la distribución y cualquier otro componente adicional que fuese necesario. Conectamos a naxtaro-base para ejecutar `lb clean` que inicializará el entorno de trabajo, de esta forma nos aseguramos que la construcción partirá de un entorno “limpio”

```
root@naxtaro-base:~/naxtaro# lb clean
[2016-05-20 18:23:32] lb clean
P: Executing auto/clean script.
[2016-05-20 18:23:32] lb clean noauto
P: Cleaning chroot
```

Ahora ya podemos proceder con el comando `lb build`:

```
root@naxtaro-base:~/naxtaro# lb build
[2016-05-20 18:25:39] lb build
P: Automatically populating config tree.
[2016-05-20 18:25:39] lb config
P: Executing auto/config script.
[2016-05-20 18:25:39] lb config noauto --binary-images iso --debian-
installer live --distribution jessie --archive-areas main contrib non-
free --bootappend-live boot=live config username=naxtaro
hostname=naxtaro --mirror-bootstrap http://localhost:3142/debian/ --
mirror-binary http://localhost:3142/debian/
P: Updating config tree for a debian/jessie/amd64 system
P: Symlinking hooks...
P: Executing auto/build script.
[2016-05-20 18:25:39] lb build noauto
P: live-build 4.0.3
```

```
P: Building config tree for a debian/jessie/amd64 system
[2016-05-20 18:25:40] lb bootstrap
P: Setting up cleanup function
[2016-05-20 18:25:40] lb bootstrap_cache restore
P: Restoring bootstrap stage from cache...
[2016-05-20 18:25:43] lb bootstrap_cdebootstrap
[2016-05-20 18:25:43] lb bootstrap_debootstrap
P: Begin bootstrapping system...
W: skipping bootstrap, already done
[2016-05-20 18:25:43] lb bootstrap_archive-keys
. . .
. . .
[2016-05-20 18:38:29] lb chroot_devpts remove
P: Begin unmounting /dev/pts...
P: Begin unmounting filesystems...
P: Saving caches...
Reading package lists...
Building dependency tree...
Reading state information...
[2016-05-20 18:38:29] lb source
root@naxtaro-base:~/naxtaro#
```

Por consola irán apareciendo diferentes mensajes que corresponderán a las distintas etapas por las que va pasando *lb build* hasta completar la imagen. El proceso dura aproximadamente unos 13 minutos, todos los detalles del proceso los tenemos examinando el fichero `build.log` que genera tal como lo definimos en el apartado 4.2.2., si todo ha ido bien pasado este tiempo tendremos en el directorio de trabajo los ficheros:

```
root@naxtaro-base:~/naxtaro# ls -l
total 424568
drwxr-xr-x  2 root root      4096 May  2 18:57 auto
drwxr-xr-x 10 root root      4096 May 20 18:38 binary
-rw-r--r--  1 root root    250858 May 20 18:38 build.log
drwxr-xr-x 10 root root      4096 May 20 18:31 cache
drwxr-xr-x 22 root root      4096 May 20 18:38 chroot
-rw-r--r--  1 root root   2028926 May 20 18:30 chroot.files
-rw-r--r--  1 root root      7347 May 20 18:27 chroot.packages.install
-rw-r--r--  1 root root     11246 May 20 18:30 chroot.packages.live
drwxr-xr-x 19 root root      4096 May 20 18:25 config
-rw-r--r--  1 root root     26357 May 20 18:38 live-image-amd64.contents
-rw-r--r--  1 root root   2028926 May 20 18:37 live-image-amd64.files
-rw-r--r--  1 root root 428881920 May 20 18:38 live-image-amd64.iso
-rw-r--r--  1 root root   1466117 May 20 18:38 live-image-amd64.iso.zsync
-rw-r--r--  1 root root     11246 May 20 18:37 live-image-amd64.packages
```

Para finalizar se renombra el nombre de la imagen creada iso,

```
root@naxtaro-base:~/naxtaro# mv live-image-amd64.iso naxtaro-linux-amd64.iso
```

Ahora ya estará lista para transferir a un servidor, a un cd-rom, a un dispositivo USB, etc.

8. Resultados y análisis

Una vez configurados todos los componentes y con la distribución ya preparada, nos quedará comprobar mediante una instalación en entorno servidor que todo funciona según lo previsto, sobretodo que la distribución se crea correctamente, se puede instalar y las variables de los ficheros de configuración son calculadas correctamente. Para ello se procede a realizar pruebas de instalación en el entorno de trabajo que se ha estado utilizando a lo largo de todo el trabajo de *Virtualbox* y otra instalación nueva en entorno *Xen*. Una vez completada la instalación se comprobará mediante algunos clientes que los servicios definidos funcionan correctamente.

8.1. Instalación en entornos virtuales

Se elige utilizar *VirtualBox* ya que es el software de gestión de máquinas virtuales que ya se ha utilizado a lo largo de todo el proceso para probar una y otra vez como iba quedando la distribución. La creación de la máquina virtual será igual que las veces anteriores pero esta vez tendremos que añadir un segundo dispositivo de red con tal de poder simular una red externa y otra interna.

8.1.1. Instalación en entorno VirtualBox

Procederemos a probar el proceso de instalación de la distribución creada, para ello ejecutamos la aplicación *Virtualbox*, igual que en apartado 4.2., seleccionaremos New, y completaremos la información:

Name: naxtaro
Type: Linux
Version: Debian (64-bit)
Memory Size: 768MB que nos mostrara por default es correcto,
Create a virtual hard disk now, la opcion recomendada de 8gb es correcta,
el tipo VDI (*virtualbox disk image*), opcion *Dynamically allocated*, y finalmente *Create*

Antes de instalar la distribución en esta nueva máquina virtual, hemos de configurar dos dispositivos de red, para poder probar todas las funciones

anteriormente definidas, para ello en las propiedades de esta nueva máquina virtual vamos a network y activaremos una segunda interfaz de red con la opción “Enable Network Adapter”

Pulsamos en *Start* seleccionamos el fichero `naxtaro-linux-amd64.iso` y comenzará el proceso de ejecución de la distribución en el entorno virtual, apareciendo inmediatamente el menú de arranque, seleccionamos la opción “*Graphical install*”

Tal y como lo configuramos en el apartado 4.5. el programa de instalación solamente nos preguntará sobre el lenguaje que queremos usar, cuál es nuestro país y el tipo de teclado, que por defecto estará en la configuración Inglesa. El resto de opciones vendrán definidas de forma automática gracias al fichero creado `preseed.cfg`. Recordemos que en este fichero de configuración estaban comentadas justamente estas opciones, para que el programa de instalación no pregunte ni siquiera estos datos, bastará con descomentar estas dos instrucciones del fichero `preseed.cfg`:

```
# Preseeding only locale sets language, country and locale.
#d-i debian-installer/locale string en_US
# Keyboard selection.
#d-i keyboard-configuration/xkb-keymap select us
```

Una vez completada esta información inicial el programa de instalación proseguirá mediante una barra indicando su avance de forma totalmente automática.

Finalizado el proceso de instalación se producirá un reinicio del nuevo servidor, mostrando al terminar la pantalla de inicio que preguntará por un usuario del sistema con el mensaje "`naxtaro login:`". Probaremos las credenciales definidas en el `preseed.cfg` que eran usuario `naxtaro` contraseña `naxtaro10`.

Comprobamos que podemos ingresar en el sistema sin problemas, el usuario naxtaro será un usuario normal del sistema sin privilegios de administración ni configuración. Para poder ejecutar el script de configuración de red de naxtaro, `naxtaro_setup.sh`, será necesario utilizar el usuario

privilegiado *root* que posee todos los permisos. Para ello ejecutaremos el comando `sudo bash` y pasaremos a tener privilegios de *root* y podremos ejecutar el script `naxtaro_setup.sh`.

```
root@naxtaro:/home/naxtaro# naxtaro_setup.sh
Welcome to Naxtaro Linux v1.0 setup
Please make sure all the ethernet cables are connected to the right devices,
eth1 card to external network and eth0 attached to internal network
============================
= External network setup =
============================
External IP address (1.1.1.10)? 80.10.10.10
External mask (255.255.255.0)?
External default gateway (1.1.1.1)? 80.10.10.1
External domain name (naxtaro.com)?
============================
= Internal network setup =
============================
Internal IP address (10.15.13.1)?
Internal mask (255.255.255.0)?
Internal domain name (naxtaroint.com)?
Setting network, please wait...
[ ok ] Restarting networking (via systemctl): networking.service.
```

Introducimos por ejemplo la dirección IP externa 80.10.10.10 y su puerta de enlace 80.10.10.1, el resto de datos los que nos muestra por defecto ya va bien por lo tanto con pulsar *Enter* tomará el valor por defecto del interior de los paréntesis. Con estos datos el script será capaz de calcular el resto de datos que necesita para completar con estas variables los ficheros de configuración necesarios. A continuación procederá a reiniciar los diferentes servicios para que los cambios surjan efecto. Sino se observa ningún mensaje de error es que todo se ha configurado correctamente.

8.1.2. Instalación en entorno Xen

Al igual que *VirtualBox*, Xen[6] es una herramienta de virtualización distribuida bajo licencia General Public License de GNU (GPL) que nos permite ejecutar varios sistemas operativos en un mismo sistema anfitrión en el cual el aislamiento entre los recursos asignados a cada sistema operativo es total. Cada uno de estos sistemas operativos residen en una máquina virtual. Xen tiene dos modos de operar: HVM o 'Full Virtualization', es decir, virtualización completa, que consiste en la instalación de un *domU* como si fuera un host independiente; o la paravirtualización, que consiste en utilizar un kernel modificado para que pueda comunicarse con el hypervisor de Xen. El usar HVM tiene la ventaja de que se puede virtualizar Windows, y que no es necesario tener un kernel especial para virtualizar sistemas GNU/Linux, pero su rendimiento es inferior, ya que la ausencia de un kernel adaptado, ciertos componentes son emulados. En nuestro caso al instalar el servidor Naxtaro usaremos virtualización HVM ya que necesitamos este modo para poder arrancar desde la imagen cd-rom y una vez instalado en el volumen local pasaremos a modo de virtualización completa. Con Xen es posible modificar el tamaño de memoria RAM asignada, conectar tarjetas de red y agregar discos en caliente. La configuración se realiza mediante un programa cliente instalado en el host, pero puede conectarse a la máquina virtual desde un cliente remoto,

es decir que cuando tengamos lista la instalación de Naxtaro podremos conectar desde el resto de la red sin problemas.

A diferencia de *VirtualBox*, la interfaz gráfica y la integración de ingreso y salida de datos es un poco precaria. Utiliza una variación de VNC[23] para el control de consola pero con un cliente *vnc* podremos conectar a la consola sin mayor complicación. VNC son las siglas en inglés de *Virtual Network Computing* (Computación Virtual en Red). VNC es un programa de software libre basado en una estructura cliente-servidor el cual permite tomar el control del ordenador servidor remotamente a través de un ordenador cliente. También llamado software de escritorio remoto.

Damos por hecho que tenemos ya configurado y listo el servidor anfitrión de Xen versión 4.4, con sus utilidades y todo lo necesario instalado. También consideraremos que este servidor tiene disponibles las utilidades de LVM[24]. LVM es el acrónimo de *Logical volume managent*, que es una forma de asignar espacio de forma más flexible que las formas tradicionales como el particionado. En particular este *volume manager* puede concatenar, dividir o combinar particiones (incluso de discos distintos) en otras virtuales más grandes que los administradores pueden redimensionar o mover potencialmente sin ni siquiera interrumpir su uso. Para crear este tipo de volúmenes se utiliza el comando `lvcreate`, en nuestro caso con la sintaxis, `lvcreate -L8G -n naxtaro-disk vg1`, que significa que queremos crear un nuevo volumen de 8Gb de tamaño, llamado “naxtaro-disk” y que lo queremos crear en un grupo de volumen llamado *vg1*. Si el comando es correcto obtendremos el mensaje: “`Logical volume "naxtaro-disk" created`”. Ya tenemos el disco preparado, obviamente también copiamos la distribución en formato *iso* al servidor Xen. El siguiente paso será crear un fichero de configuración dentro del directorio `/etc/xen` que llamaremos `naxtaro-install.cfg` con el contenido:

```
builder='hvm'

vcpus       = '1'
memory      = '1024'

root        = '/dev/xvda1 ro'
disk        = [
              'file:/root/naxtaro-linux-amd64.iso,ioemu:hdc:cdrom,r',
              'phy:/dev/vg1/naxtaro-disk,xvda,w',
              ]

name        = 'naxtaro-install'

vif         = [ 'ip=192.168.1.150,bridge=xenbr1',
                'ip=10.15.13.1,bridge=xenbr0' ]
boot="d"

vnc=1
vnclisten="0.0.0.0"
```

```
vncdisplay=1
vncpasswd="naxtaro10"

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Especificamos que se usará *hvm* como método de virtualización, se asignará una *cpu* virtual, se usará 1Gb de memoria *ram*, como disco virtuales definimos la imagen *iso* creada de la distribución y el volumen de 8gb que hemos creado anteriormente. El nombre del servidor será “naxtaro-install”, definimos dos dispositivos de red con sus correspondientes direcciones IPs, especificamos que con la instrucción “boot=d” que queremos que arranque desde la unidad de cd-rom, y que queremos usar el protocolo *vnc* para conectar al servidor con password “naxtaro10” y en el *display* 1 que equivale al puerto 5901. La última sección hace referencia a como queremos que responda la maquina virtual cuando la apagamos, cuando la reiniciamos o cuando por algún error no responda.

Una vez todo listo podemos proceder a arrancar la máquina virtual invocando la utilidad *xl* que es la herramienta que usa Xen para gestionar los dominios o máquinas virtuales. Gracias a esta utilidad podemos crear, parar o reiniciar o obtener una lista de estos dominios. Para arrancar esta instancia de instalación de Naxtaro desde consola escribiremos la instrucción:

```
root@nax-xen:/etc/xen# xl create naxtaro-install.cfg
Parsing config from naxtaro-install.cfg
root@spf-pre:/etc/xen#
```

Ya tendremos la instancia en marcha, para conectar desde un cliente *vnc* especificaríamos el nombre del host donde hemos arrancado la máquina virtual, puerto 5901 y el password asignado anteriormente “naxtaro10”. Tendremos en pantalla la misma imagen de instalación que tuvimos en el apartado anterior con VirtualBox. El proceso de instalación será exactamente igual, solamente nos preguntará sobre el lenguaje que queremos usar, cuál es nuestro país y el tipo de teclado, que por defecto estará en la configuración Inglesa. El resto se completará automáticamente.

En este punto ya tendremos la distribución instalada, ahora debemos modificar el fichero de configuración Xen y utilizar virtualización total ya que no necesitaremos arrancar desde el cd-rom. Crearemos el fichero `naxtaro.cfg` con la configuración:

```
bootloader = '/usr/lib/xen-4.4/bin/pygrub'

vcpus       = '1'
memory      = '1024'

root        = '/dev/xvda1 ro'
```

```
disk         = [
              'phy:/dev/vg1/naxtaro-disk,xvda,w'
               ]

name         = 'naxtaro'

vif          = [ 'ip=192.168.1.150,bridge=xenbr1',
                 'ip=10.15.13.1,bridge=xenbr0' ]

on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

El fichero de configuración se ha modificado ya que ya no necesitaremos arrancar desde la imagen *iso* ya que la distribución ya se encuentra instalada en el volumen `/dev/vg1/naxtaro-disk` y ya no necesitamos el protocolo VNC para conectar ya que podremos hacerlo directamente por red como veremos en el apartado siguiente.

8.2. Pruebas básicas de servicios

Ya podemos proceder a realizar una serie de pruebas básicas para comprobar que la distribución esta correctamente instalada y los servicios que hemos definido funcionan. Empezaremos por el servicio de *ssh*, la prueba es realmente simple hay que realizar una conexión *ssh* con la ip del servidor, especificando el usuario “naxtaro”:

```
[18:48] nax@callahan:~ > ssh naxtaro@192.168.1.150
The authenticity of host '192.168.1.150 (192.168.1.150)' can't be established.
ECDSA key fingerprint is SHA256:dKPtRbxfeP1IKEYl1NMvCzfolQuosN8XcPJuAN9G6k8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.150' (ECDSA) to the list of known hosts.
naxtaro@192.168.1.150's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed May 25 16:47:51 2016 from 192.168.1.39
naxtaro@naxtaro:~$
```

Para el servicio de *webproxy*, *squid*, podemos invocar la utilidad *wget*[25], que es una herramienta libre que permite la descarga de contenidos desde servidores web de una forma simple. Su nombre deriva de World Wide Web (w), y de “obtener” (en inglés *get*), esto quiere decir: obtener desde la WWW. Mediante los parámetros de usar proxy y la dirección del proxy y el puerto, en nuestro caso *localhost* 3128 haremos la prueba de bajarnos del servidor de google.com el fichero html de su página principal.

```
root@naxtaro:~# wget -e use_proxy=yes -e http_proxy=localhost:3128 google.com
--2016-05-25 16:55:43--  http://google.com/
Resolving localhost (localhost)... ::1, 127.0.0.1
Connecting to localhost (localhost)|::1|:3128... connected.
Proxy request sent, awaiting response... 302 Found
Location: http://www.google.es/?gfe_rd=cr&ei=VNhFV-X2OITY8gf0rqDQAw [following]
--2016-05-25 16:55:43--  http://www.google.es/?gfe_rd=cr&ei=VNhFV-X2OITY8gf0rqDQAw
Reusing existing connection to [localhost]:3128.
Proxy request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'index.html'

index.html                              [ <=>

2016-05-25 16:55:43 (15.4 MB/s) - 'index.html' saved [11611]
```

De forma similar procederemos a probar que el sistema utiliza el cache de *apt* que instalamos, *apt-cacher-ng* simplemente ejecutando el comando `apt-get update` veremos por pantalla que esta conectando a *localhost* y el puerto 3142 que corresponde a este servicio:

```
root@naxtaro:~# apt-get update
Get:1 http://security.debian.org jessie/updates InRelease [63.1 kB]
Ign http://localhost:3142 jessie InRelease
Get:2 http://security.debian.org jessie/updates/main Sources [136 kB]
Ign http://localhost:3142 stable InRelease
Get:3 http://localhost:3142 jessie-updates InRelease [142 kB]
Get:4 http://security.debian.org jessie/updates/contrib Sources [1,439 B]
Get:5 http://localhost:3142 jessie Release.gpg [2,373 B]
Get:6 http://localhost:3142 stable Release.gpg [2,373 B]
Get:7 http://security.debian.org jessie/updates/non-free Sources [14 B]
Get:8 http://localhost:3142 jessie Release [148 kB]
Get:9 http://security.debian.org jessie/updates/main amd64 Packages [247 kB]
Get:10 http://localhost:3142 jessie-updates/main Sources [14.1 kB]
Get:11 http://security.debian.org jessie/updates/contrib amd64 Packages [2,506 B]
Get:12 http://localhost:3142 jessie-updates/main Translation-en [9,364 B]
Get:13 http://localhost:3142 stable Release [148 kB]
Get:14 http://security.debian.org jessie/updates/non-free amd64 Packages [14 B]
Get:15 http://localhost:3142 jessie/main amd64 Packages [6,763 kB]
Get:16 http://security.debian.org jessie/updates/contrib Translation-en [1,211 B]
Get:17 http://security.debian.org jessie/updates/main Translation-en [135 kB]
Get:18 http://security.debian.org jessie/updates/non-free Translation-en [14 B]
Get:19 http://localhost:3142 jessie/main Translation-en [4,581 kB]
Get:20 http://localhost:3142 jessie-updates/main amd64 Packages [8,456 B]
Get:21 http://localhost:3142 stable/main Sources [7,059 kB]
Get:22 http://localhost:3142 stable/contrib Sources [50.7 kB]
Get:23 http://localhost:3142 stable/non-free Sources [99.2 kB]
Fetched 19.6 MB in 8s (2,442 kB/s)
Reading package lists... Done
```

Ahora seria el turno de comprobar que las carpetas compartidas que definimos en la configuración de samba son accesibles, para ello se utilizará la herramienta *smbclient*[26] que es básicamente un cliente de samba que permite examinar y montar discos remotos compartidos en un servidor Linux.

Se uso será de la forma mas básica, mediante el parámetro -L y nombre del servidor nos mostrará las carpetas que ese servidor esta compartiendo:

```
root@naxtaro:~# smbclient -L localhost
Enter root's password:
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.2.10-Debian]

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        datos           Disk      datos
        misc            Disk      miscellaneous
        IPC$            IPC       IPC Service (Samba 4.2.10-Debian)
Domain=[WORKGROUP] OS=[Windows 6.1] Server=[Samba 4.2.10-Debian]

        Server               Comment
        ---------            -------
        NAXTARO              Samba 4.2.10-Debian
        NAXTARO-BASE         Samba 4.2.10-Debian

        Workgroup            Master
        ---------            -------
        WORKGROUP            NAXTARO
root@naxtaro:~#
```

Comprobamos que efectivamente en la lista aparecen “datos” y “misc” aparte de las impresoras que también se configuraron en pasado apartado 5.10.

El siguiente seria probar a conectar al servicio de base de datos, MySQL, con el usuario y password definidos anteriormente:

```
root@naxtaro:~# mysql -uroot -pnaxtaro2016
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.47-0+deb8u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

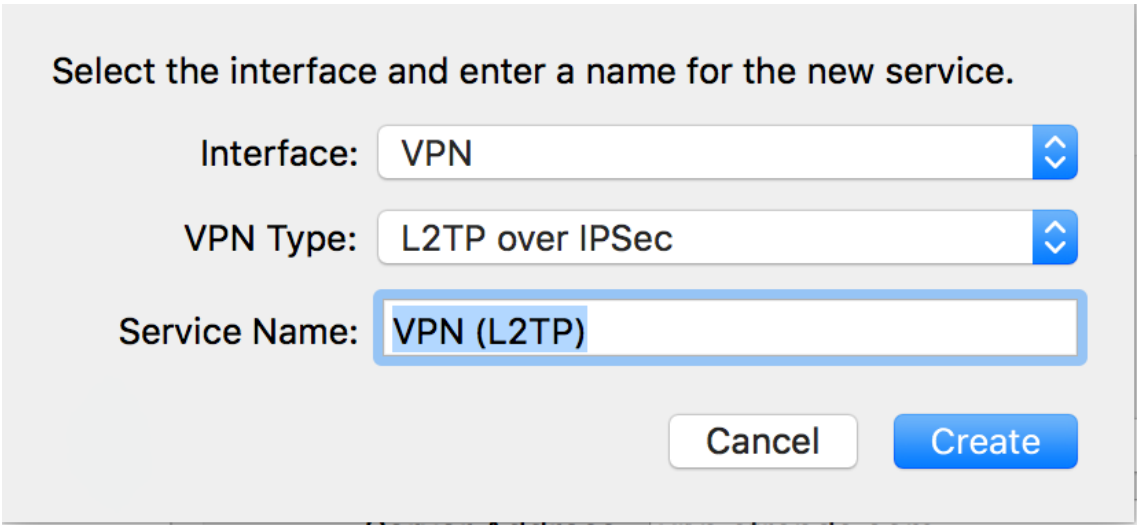
Para verificar el servicio de resolución de nombres DNS, *bind9*, utilizaremos la utilidad del sistema `nslookup` (“Name System Lookup”, en inglés) que es una herramienta que permite consultar un servidor de nombres y obtener información relacionada con el dominio o el host y así diagnosticar los eventuales problemas de configuración que pudieran haber surgido en el DNS.

Por lo tanto usaremos esta utilidad para resolver los nombres de hosts que definimos y que efectivamente se traducen en las direcciones IP que definimos:

Como vemos los resultados son los esperados, los hosts de los dos dominios que definimos en el apartado 5.6. naxtaro.com y naxtaroint.com

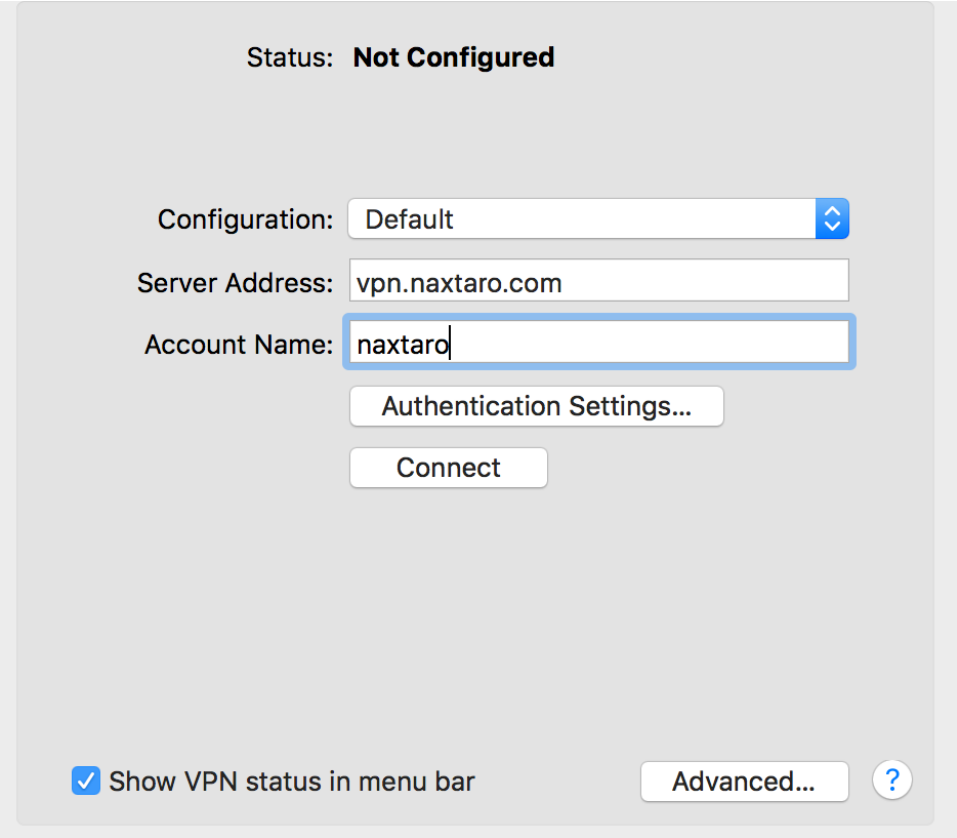
resuelven con las direcciones IP correctas, mail.naxtaro.com con la 3.3.3.3, www.naxtaro.com con la 4.4.4.4, para el dominio interno, tendríamos otro www.naxtaroint.com con IP 5.5.5.5 y un supuesto servidor de git, que respondería a la IP 6.6.6.6.

Para la prueba del servicio de VPN vamos a configurar un cliente *mac* que ya trae incluido en su sistema operativo el cliente VPN con la variante L2TP sobre IPSec, para ello simplemente en las Preferencias del Sistema, Red, creamos una nueva conexión, interfaz VPN, del tipo L2TP sobre IPSec, y en el nombre del servicio lo podemos dejar tal cual:

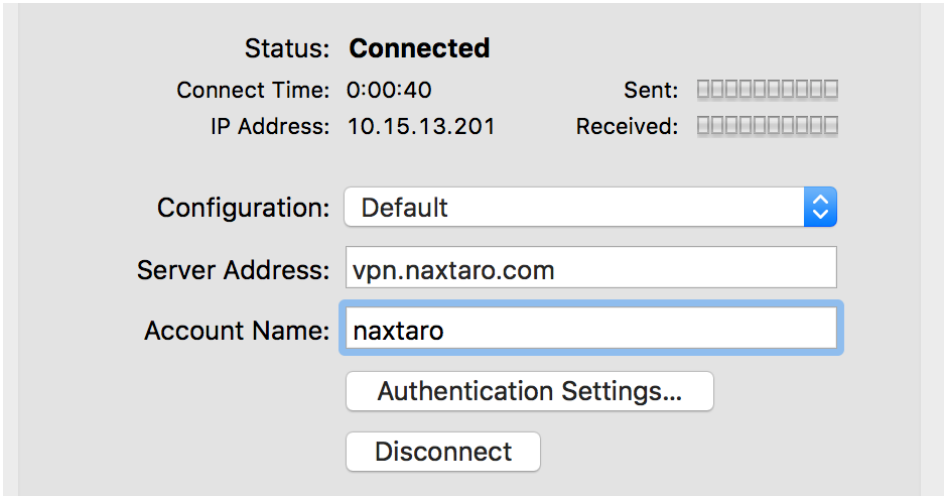


pulsamos en Create y en el campo dirección del servidor pondremos el nombre del servidor VPN, en nuestro caso vpn.naxtaro.com, como nombre de usuario el del sistema “naxtaro” ya que la autenticación ya la definimos de esta manera

en el apartado 5.7. En valores de autenticación introduciremos la contraseña del usuario “naxtaro”, que recordemos es “naxtaro10” y finalmente en secreto



compartido la contraseña que definimos en el fichero `ipsec.conf` que es “ClientsSuperStrongPassword2016”. Con estos simples pasos ya tendriamos el cliente listo y configurado para conectar a nuestro servidor de VPN, pulsamos sobre “Connect” y si todo va bien nos aparecerá la siguiente ventana que nos confirma que estamos conectados, que IP local nos ha asignado (10.15.13.201) y paquetes enviados y/o recibidos.



Para el servicio de NTP emplearemos la utilidad cliente *ntpdate* que mediante el parámetro especificará la IP del servidor Naxtaro, en este caso 192.168.1.150:

```
root@naxtaro-base:~# ntpdate 192.168.1.150
27 May 22:50:29 ntpdate[26902]: adjust time server 192.168.1.150 offset -0.003563 sec
root@naxtaro-base:~#
```

Mediante el comando `/etc/init.d/fw status` podemos visualizar el tipo de reglas, a que dispositivo de red afectan puertos abiertos etc.

```
root@naxtaro:~# /etc/init.d/fw status
Displaying rules...
Chain INPUT (policy DROP 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   52  3570 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
 3790  639K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW,ESTABLISHED
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           tcp spt:22 state ESTABLISHED
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           multiport sports 80,443 state ESTABLISHED
    0     0 ACCEPT     udp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           multiport sports 53,123
    0     0 ACCEPT     tcp  --  eth1   *       0.0.0.0/0            0.0.0.0/0           multiport sports 25,443 state ESTABLISHED
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0           tcp dpt:4500
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:1701
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:4500
    0     0 ACCEPT     udp  --  *      *       0.0.0.0/0            0.0.0.0/0           udp dpt:500
    0     0 ACCEPT     esp  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
   52  3570 ACCEPT     all  --  *      lo      0.0.0.0/0            0.0.0.0/0
 2723  291K ACCEPT     all  --  *      *       0.0.0.0/0            0.0.0.0/0           state NEW,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           tcp dpt:22 state NEW,ESTABLISHED
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           multiport dports 80,443 state NEW,ESTABLISHED
    0     0 ACCEPT     udp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           multiport dports 53,123
    0     0 ACCEPT     tcp  --  *      eth1    0.0.0.0/0            0.0.0.0/0           multiport dports 25,443 state NEW,ESTABLISHED

Chain FORWARD_DROP (0 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "FORWARD... "
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain INPUT_DROP (0 references)
 pkts bytes target     prot opt in     out     source               destination
    0     0 LOG        all  --  *      *       0.0.0.0/0            0.0.0.0/0           LOG flags 0 level 4 prefix "INPUT... "
    0     0 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0
```

Finalmente para el servicio de alta disponibilidad, definido e instalado en el anterior apartado 5.11, al tener solamente un nodo instalado obviamente las pruebas serán muy limitadas, pero podemos comprobar mediante los mensajes del sistema que el servicio se inicia correctamente

```
May 25 17:21:58 naxtaro heartbeat: [5553]: info: Enabling logging daemon
May 25 17:21:58 naxtaro heartbeat: [5553]: info: logfile and debug file are those specified in logd config file (default /etc/logd.cf)
May 25 17:21:58 naxtaro heartbeat: [5553]: info: Pacemaker support: false
May 25 17:21:58 naxtaro heartbeat: [5553]: info: **************************
May 25 17:21:58 naxtaro heartbeat: [5553]: info: Configuration validated. Starting heartbeat 3.0.5
May 25 17:21:58 naxtaro heartbeat[5395]: Done.
May 25 17:21:58 naxtaro heartbeat: [5554]: info: heartbeat: version 3.0.5
May 25 17:21:58 naxtaro heartbeat: [5554]: info: Heartbeat generation: 1464196758
May 25 17:21:58 naxtaro heartbeat: [5554]: info: glib: UDP Broadcast heartbeat started on port 2694 (2694) interface eth0
May 25 17:21:58 naxtaro heartbeat: [5554]: info: glib: UDP Broadcast heartbeat closed on port 2694 interface eth0 - Status: 1
May 25 17:21:58 naxtaro heartbeat: [5554]: info: Local status now set to: 'up'
. ok
root@naxtaro:/etc/ha.d# May 25 17:21:59 naxtaro heartbeat: [5554]: info: Link naxtaro:eth0 up.
```

Podemos leer que el servicio *heartbeat* arranca correctamente, indicando su versión y que usara el interface de red *eth0* y el puerto número 2694 tipo *udp* para comunicarse con otros servidores.

9. Conclusiones

Como conclusiones finales destacar que ha sido un Proyecto muy intenso y satisfactorio, aunque muchos de los servicios aquí descritos personalmente ya los había implementado en más de una ocasión, la parte de creación de la version Live e instalable, así como el desarrollo de *shellscripts* y configuraciones para crear una instalación prácticamente automática ha sido la parte que he encontrado más interesante.

Gracias a mi dilatada experiencia como Administrador de Sistemas he podido seleccionar y probar a lo largo de varios años diferentes herramientas y servicios de GNU/Linux que luego he podido adaptar a este Proyecto. Ha sido justamente esta parte de implementación a nivel práctico y en entornos reales lo que ha hecho el trabajo más ameno e interesante.

La planificación inicial no ha sido la real a lo largo de la elaboración del Proyecto ya que algunas partes llevaron más tiempo de lo previsto, provocado que el resto de fases se viesen afectadas. El tiempo de creación de cada imagen *iso* y su posterior instalación en *Virtualbox* era un proceso largo que tenía que esperar a su finalización para comprobar los resultados. Además el documento no sigue un orden cronológico acorde a los capítulos, por ejemplo la parte de automatización de la instalación no la elaboré casi al final del proceso, aunque esté reflejado en esta Memoria al principio de ella, obligándome a reordenar y dedicar más tiempo a esa parte que no tenia prevista inicialmente. Sin lugar a dudas las diferentes entregas en forma de PACs son fundamentales para la planificación y realización puntual del Proyecto ya que obligan en mayor medida a ir avanzando y no dejar para más adelante aspectos clave que luego no podamos abordar por falta de tiempo y recursos.

Con respecto a los resultados obtenidos en el proyecto, indicar que creo que se han cumplido cada uno de los objetivos que se habían propuesto en un primer momento, ya que se ha obtenido con éxito una distribución personalizada y a mi entender muy fácil de configurar y modificar a gusto de cada Administrador de Sistemas. Me hubiese gustado poder incluir algún servicio adicional que considero también de importancia como hubiese sido un servidor de correo saliente y entrante implementado bajo un protocolo seguro SSL o también un servidor web con soporte https pero lamentablemente la falta de tiempo y el no hacer demasiado extenso el presente Documento me he decantado finalmente a descartarlos. También hay que tener en cuenta que cuantos más servicios añadamos a nuestra distribución, más vulnerable podría ser nuestro sistema. No obstante, uno de los objetivos principales de Naxtaro que es la de cortafuegos o firewall se ha cubierto en su totalidad.

Aunque las funcionalidades definidas se han cubierto, me hubiese gustado poder implementar un sistema gráfico más vistoso que un *shellscript* para la instalación y la recopilación de datos, o incluso ir más allá e incluir una interfaz web (https) de administración y configuración de los servicios, un directorio de LDAP o similar para la gestión de usuarios, incluir mas características de seguridad como algún software de prevención de intrusos (IDS), antivirus, prevención de *spyware*, control del ancho y uso de la red, monitorización de los servicios y envío de alertas en caso de fallo, etc,

Temas pendientes para un probable futuro Naxtaro v2.0:

- Asistente de configuración de datos de red iniciales en modo gráfico, migrar `naxtaro_setup.sh`.
- Acceso por https a la consola de administración donde se pueda consultar el estado y parar/arrancar servicios, configurar reglas del firewall de manera cómoda y sencilla, estadísticas de uso, etc
- Monitorización de la red, estadísticas sobre el ancho de banda usado, paquetes recibidos/enviados, etc.
- Monitorización a nivel de hardware, chequeo del espacio en disco duro, memoria utilizada/libre, uso de la cpu, envio de alertas por mail o sms.
- Implementación de un sistema IDS (*Intrusion Detection System*), informes de seguridad de la red, accesos no autorizados a puertos o servicios, intentos de conexión no permitidos, etc.
- Opción de bloqueo por geolocalización, poder bloquear accesos provenientes de según que país.
- QoS (*Quality Of Service*): poder definir en Naxtaro que tipo de tráfico tiene prioridad sobre otros y que ancho de banda podemos asignar para garantizar que no se sature la red, poder priorizar por ejemplo el tipo de tráfico web o *voip* antes que el *p2p*.
- Incluir algún servicio adicional importante como el de servidor de mail entrante y saliente.
- Cliente que de manera automática según un periodo establecido chequee y actualize a la última versión desde red, sin necesidad de tener que bajar la imagen iso entera y tener que proceder a su instalación de nuevo etc

10. Glosario

Debian: Sistema Operativo Linux robusto, fácil de instalar y de administrar gracias al amplio catalogo de software disponible en forma de paquetes que se gestionan de manera muy sencilla.

Debian Live Project: proyecto que usa la distribución Debian de forma oficial para generar sus versiones Live.

Bash: (Bourne again shell) consola de comandos Unix escrita para el proyecto GNU, cuya función consiste en interpretar órdenes, y un lenguaje de programación de consola. Está basado en la shell de Unix y es compatible con POSIX.

Script o *shellscript*: archivo que contiene uno más comandos del shell. Se puede crear un script a partir de un editor de textos.

Apt: (Advanced Packaging Tool) sistema de gestión de paquetes creado por el proyecto Debian. APT simplifica en gran medida la instalación y eliminación de programas en los sistemas GNU/Linux.

Apt-get: instala nuevos paquetes en un servidor Debian.

Imagen ISO: archivo informático donde se almacena una copia o imagen exacta de un sistema de archivos o ficheros de un disco óptico.

Bootstrap: Término utilizado para describir el arranque, o proceso de inicio de cualquier ordenador.

Chroot: Operación que invoca un proceso, cambiando para este y sus hijos el directorio raíz del sistema

Virtualizar: creación a través de software de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red.

Virtualbox: software que permite virtualizar varios sistemas operativos en una maquina física.

Xen: similar a Virtualbox, más orientado a servidores y sin una interfaz gráfica pero con utilidades de consola muy potentes**.**

SSH: (Secure Shell) protocolo para acceder de forma segura a servidores remotos a través de una red.

Iptables: poderoso firewall integrado en el kernel de Linux y que forma parte del proyecto *netfilter*. Iptables puede ser configurado directamente, como

también por medio de un *frontend* o una GUI. Iptables es usado por IPv4, en tanto que ip6tables es usado para IPv6

Apt-cacher: sistema de caché de repositorios.

MySQL: sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales.

DHCP: (Dynamic Host Configuration Protocol) es un estándar TCP/IP diseñado para simplificar la administración de la configuración IP de los equipos de nuestra red.

DNS: (Domain Name System) es un sistema para asignar nombres a equipos y servicios de red que se organiza en una jerarquía de dominios

Nslookup (Name System Lookup) es una herramienta que permite consultar un servidor de nombres y obtener información relacionada con el dominio o el host y así diagnosticar los eventuales problemas de configuración que pudieran haber surgido en el servidor DNS.

VPN (Virtual Private Network) permite crear una conexión segura a otra red a través del Internet.

Site2site: termino que refiere a la interconexión a través de una VPN de 2 localizaciones con redes distintas.

IPsec: (Internet Protocol Security) protocolo que permite mejorar la seguridad a través de algoritmos de cifrado robustos y un sistema de autenticación más exhaustivo

PPP: Point-to-Point Protocol, es un protocolo de nivel de enlace de datos, que provee autenticación de conexión, cifrado de transmisión y compresión.

NTP: (Network Time Protocol) permite sincronizar los relojes de los dispositivos que lo consulten.

Webproxy: aplicación que nos permitirá mantener una copia local del contenido más frecuente que visiten los diferentes dispositivos de nuestra red local.

Squid: software que implementa una solución Webproxy.

Samba: software que implementa una solución de comparición de directorios y ficheros en red.

Smbclient: cliente de samba que permite examinar y montar discos remotos de forma local.

HA (High Availability): Protocolo de diseño del sistema y su implementación asociada que asegura un cierto grado absoluto de continuidad operacional durante un período de medición dado. Disponibilidad se refiere a la habilidad de la comunidad de usuarios para acceder al sistema, someter nuevos trabajos, actualizar o alterar trabajos existentes o recoger los resultados de trabajos previos. Si un usuario no puede acceder al sistema se dice que está no disponible. El término tiempo de inactividad (downtime) es usado para definir cuándo el sistema no está disponible.

Clúster: se aplica a los conjuntos o conglomerados de ordenadores unidos entre sí normalmente por una red de alta velocidad y que se comportan como si fuesen una única computadora.

11. Bibliografía

[1] https://www.debian.org
[2] https://www.virtualbox.org

[3] https://www.debian.org/intro/why_debian
[4] https://www.debian.org/releases/

[5] https://lists.debian.org/debian-live/

[6] http://wiki.xen.org/wiki/Xen_Project_Software_Overview

[7] https://www.debian.org/devel/debian-live/

[8] https://www.unix-ag.uni-kl.de/~bloch/acng/

[9] https://www.debian.org/releases/stable/example-preseed.txt

[10] http://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/

[11] https://serversforhackers.com/video/installing-mysql-with-debconf

[12] http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m2/servidor_dhcp.html

[13] https://msdn.microsoft.com/es-es/library/cc787920(v=ws.10).aspx
[14] http://www.gpltarragona.org/archives/421

[15] http://www.welivesecurity.com/la-es/2012/09/10/vpn-funcionamiento-privacidad-informacion
[16] https://wiki.strongswan.org/projects/strongswan/wiki/IntroductionTostrongSwan
[17] https://wiki.gentoo.org/wiki/IPsec_L2TP_VPN_server
[17] https://wiki.openwrt.org/inbox/openswanxl2tpvpn

[18] http://www.thegeekstuff.com/2014/06/linux-ntp-server-client/

[19] http://linuxaria.com/pills/how-to-setup-a-squid-proxy-on-your-debian-linux

[20] https://wiki.debian.org/SambaServerSimple

[21] http://www.linux-ha.org/doc/users-guide/users-guide.html

[22] https://debian-live.alioth.debian.org/live-manual/stable/manual/html/live-manual.en.html#616

[23] https://es.wikipedia.org/wiki/VNC

[24] https://glatelier.org/2010/01/19/lvm-en-gnulinux-parte-i-que-es-y-a-quien-le-sirve/

[25] http://stackoverflow.com/questions/11211705/setting-proxy-in-wget

[26] http://personales.upv.es/~resteban/sambaupv/node8.html

12. Anexos

10.1. Coste del servidor

Un posible servidor para Naxtaro seria algo del tipo como el de la imagen pero teniendo en cuenta que deberíamos adquirir una segunda tarjeta de red

que podemos encontrar por ejemplo en amazon.com por 7,89€ (https://www.amazon.es/D-Link-DGE-528T-Tarjeta-Gigabit-Ethernet/dp/B0006I02VI/ref=sr_1_1?ie=UTF8&qid=1464639457&sr=8-1). En total por 62€ tendríamos un hardware perfectamente válido y operativo para poder instalar Naxtaro. Se trata de una inversión irrisoria a cambio de los servicios que ofrece, gracias a poder emplear justamente herramientas GNU/Linux para la parte del sistema operativo y utilidades.

12.2. Listado de ficheros

Listado del total de ficheros dentro del directorio `config` de la distribución Naxtaro:

```
root@naxtaro-base:~/naxtaro/config# ls -lR
.:
total 72
drwxr-xr-x 2 root root 4096 Mar 30 23:55 apt
drwxr-xr-x 2 root root 4096 Mar 30 23:55 archives
drwxr-xr-x 3 root root 4096 Apr 14 16:22 bootloaders
-rw-r--r-- 1 root root  199 May 26 04:27 build
drwxr-xr-x 2 root root 4096 May 19 12:09 debian-installer
drwxr-xr-x 2 root root 4096 May  9 19:03 hooks
drwxr-xr-x 2 root root 4096 Mar 30 23:55 includes
```

```
drwxr-xr-x 3 root root 4096 Apr 12 04:07 includes.binary
drwxr-xr-x 2 root root 4096 Mar 30 23:55 includes.bootstrap
drwxr-xr-x 4 root root 4096 May 18 12:41 includes.chroot
drwxr-xr-x 3 root root 4096 May 24 14:55 includes.installer
drwxr-xr-x 2 root root 4096 Mar 30 23:55 includes.source
drwxr-xr-x 2 root root 4096 May 25 18:10 package-lists
drwxr-xr-x 2 root root 4096 Mar 30 23:55 packages
drwxr-xr-x 2 root root 4096 Mar 30 23:55 packages.binary
drwxr-xr-x 2 root root 4096 Mar 30 23:55 packages.chroot
drwxr-xr-x 2 root root 4096 Mar 30 23:55 preseed
drwxr-xr-x 2 root root 4096 Mar 30 23:55 rootfs

./apt:
total 0

./archives:
total 0

./bootloaders:
total 4
drwxr-xr-x 2 root root 4096 Apr 14 20:34 isolinux

./bootloaders/isolinux:
total 68
-rw-r--r-- 1 root root   153 Apr 14 16:22 advanced.cfg
lrwxrwxrwx 1 root root    38 Apr 14 16:22 hdt.c32 -> /usr/lib/
syslinux/modules/bios/hdt.c32
-rw-r--r-- 1 root root   300 Apr 14 16:22 install.cfg
lrwxrwxrwx 1 root root    30 Apr 14 16:22 isolinux.bin -> /usr/
lib/ISOLINUX/isolinux.bin
-rw-r--r-- 1 root root    57 Apr 14 16:22 isolinux.cfg
lrwxrwxrwx 1 root root    42 Apr 14 16:22 ldlinux.c32 -> /usr/
lib/syslinux/modules/bios/ldlinux.c32
lrwxrwxrwx 1 root root    43 Apr 14 16:22 libcom32.c32 -> /usr/
lib/syslinux/modules/bios/libcom32.c32
lrwxrwxrwx 1 root root    42 Apr 14 16:22 libutil.c32 -> /usr/
lib/syslinux/modules/bios/libutil.c32
-rw-r--r-- 1 root root   252 Apr 14 16:22 live.cfg.in
-rw-r--r-- 1 root root   305 Apr 14 20:32 menu.cfg
-rw-r--r-- 1 root root 17083 Apr 14 16:22 splash.png
-rw-r--r-- 1 root root 23387 Apr 14 20:34 splash.svg
-rw-r--r-- 1 root root   508 Apr 14 16:22 stdmenu.cfg
lrwxrwxrwx 1 root root    43 Apr 14 16:22 vesamenu.c32 -> /usr/
lib/syslinux/modules/bios/vesamenu.c32

./debian-installer:
total 0

./hooks:
total 132
lrwxrwxrwx 1 root root  64 Mar 30 23:55 0010-disable-kexec-
tools.hook.chroot -> /usr/share/live/build/hooks/0010-disable-
kexec-tools.hook.chroot
lrwxrwxrwx 1 root root  64 Mar 30 23:55 0020-create-mtab-
symlink.hook.chroot -> /usr/share/live/build/hooks/0020-create-
mtab-symlink.hook.chroot
```

```
lrwxrwxrwx 1 root root  73 Mar 30 23:55 0100-remove-adjtime-
configuration.hook.chroot -> /usr/share/live/build/hooks/0100-
remove-adjtime-configuration.hook.chroot
lrwxrwxrwx 1 root root  64 Mar 30 23:55 0110-remove-backup-
files.hook.chroot -> /usr/share/live/build/hooks/0110-remove-
backup-files.hook.chroot
lrwxrwxrwx 1 root root  67 Mar 30 23:55 0120-remove-dbus-
machine-id.hook.chroot -> /usr/share/live/build/hooks/0120-
remove-dbus-machine-id.hook.chroot
lrwxrwxrwx 1 root root  68 Mar 30 23:55 0130-remove-gnome-icon-
cache.hook.chroot -> /usr/share/live/build/hooks/0130-remove-
gnome-icon-cache.hook.chroot
lrwxrwxrwx 1 root root  61 Mar 30 23:55 0140-remove-log-
files.hook.chroot -> /usr/share/live/build/hooks/0140-remove-
log-files.hook.chroot
lrwxrwxrwx 1 root root  71 Mar 30 23:55 0150-remove-mdadm-
configuration.hook.chroot -> /usr/share/live/build/hooks/0150-
remove-mdadm-configuration.hook.chroot
lrwxrwxrwx 1 root root  76 Mar 30 23:55 0160-remove-openssh-
server-host-keys.hook.chroot -> /usr/share/live/build/hooks/
0160-remove-openssh-server-host-keys.hook.
chroot
lrwxrwxrwx 1 root root  61 Mar 30 23:55 0170-remove-python-
py.hook.chroot -> /usr/share/live/build/hooks/0170-remove-
python-py.hook.chroot
lrwxrwxrwx 1 root root  70 Mar 30 23:55 0180-remove-systemd-
machine-id.hook.chroot -> /usr/share/live/build/hooks/0180-
remove-systemd-machine-id.hook.chroot
lrwxrwxrwx 1 root root  67 Mar 30 23:55 0190-remove-temporary-
files.hook.chroot -> /usr/share/live/build/hooks/0190-remove-
temporary-files.hook.chroot
lrwxrwxrwx 1 root root  69 Mar 30 23:55 0195-remove-ssl-cert-
snakeoil.hook.chroot -> /usr/share/live/build/hooks/0195-remove-
ssl-cert-snakeoil.hook.chroot
lrwxrwxrwx 1 root root  76 Mar 30 23:55 0200-remove-udev-
persistent-cd-rules.hook.chroot -> /usr/share/live/build/hooks/
0200-remove-udev-persistent-cd-rules.hook.
chroot
lrwxrwxrwx 1 root root  77 Mar 30 23:55 0300-remove-udev-
persistent-net-rules.hook.chroot -> /usr/share/live/build/hooks/
0300-remove-udev-persistent-net-rules.hoo
k.chroot
lrwxrwxrwx 1 root root  66 Mar 30 23:55 0400-update-apt-file-
cache.hook.chroot -> /usr/share/live/build/hooks/0400-update-
apt-file-cache.hook.chroot
lrwxrwxrwx 1 root root  68 Mar 30 23:55 0410-update-apt-xapian-
index.hook.chroot -> /usr/share/live/build/hooks/0410-update-
apt-xapian-index.hook.chroot
lrwxrwxrwx 1 root root  67 Mar 30 23:55 0420-update-glx-
alternative.hook.chroot -> /usr/share/live/build/hooks/0420-
update-glx-alternative.hook.chroot
lrwxrwxrwx 1 root root  68 Mar 30 23:55 0430-update-mlocate-
database.hook.chroot -> /usr/share/live/build/hooks/0430-update-
mlocate-database.hook.chroot
lrwxrwxrwx 1 root root  70 Mar 30 23:55 0440-update-nvidia-
alternative.hook.chroot -> /usr/share/live/build/hooks/0440-
update-nvidia-alternative.hook.chroot
```

```
-rwxr-xr-x 1 root root 237 May  5 12:58 0500-ssh-
server.sh.chroot
-rwxr-xr-x 1 root root 230 Apr 25 12:17 0501-apt-
cacher.sh.chroot
-rwxr-xr-x 1 root root  54 May  5 12:33 0502-dhcpd-
server.sh.chroot
-rwxr-xr-x 1 root root 255 Mar 31 04:47 0502-mysql-
server.sh.chroot
-rwxr-xr-x 1 root root  44 May  5 15:41 0504-bind9-
server.sh.chroot
-rwxr-xr-x 1 root root  98 May  6 00:12 0505-
strongswan.sh.chroot
-rwxr-xr-x 1 root root  48 May  6 13:36 0506-heartbeat.sh
-rwxr-xr-x 1 root root  42 May  9 18:58 0506-ntp.sh.chroot
-rwxr-xr-x 1 root root  45 May  6 20:15 0507-squid3.sh.chroot
-rwxr-xr-x 1 root root  44 May  9 19:03 0508-samba.sh.chroot
-rwxr-xr-x 1 root root  48 May  6 20:15 0509-heartbeat.sh.chroot
-rwxr-xr-x 1 root root 108 Apr 25 18:57 0599-misc.sh.chroot
-rwxr-xr-x 1 root root 110 Mar 31 04:48 0600-apt-
autoremove.sh.chroot

./includes:
total 0

./includes.binary:
total 4
drwxr-xr-x 2 root root 4096 Apr 12 06:15 isolinux

./includes.binary/isolinux:
total 24
-rw-r--r-- 1 root root 20541 Apr 14 14:36 splash.png

./includes.bootstrap:
total 0

./includes.chroot:
total 8
drwxr-xr-x  2 root root 4096 May 25 17:30 bin
drwxr-xr-x 12 root root 4096 May 25 17:17 etc

./includes.chroot/bin:
total 8
-rwxr-xr-x 1 root root 5386 May 25 17:30 naxtaro_setup.sh

./includes.chroot/etc:
total 52
drwxr-xr-x 2 root root 4096 May 19 13:28 apt
drwxr-xr-x 2 root root 4096 May 25 12:29 bind
drwxr-xr-x 2 root root 4096 May 25 17:11 dhcp
drwxr-xr-x 3 root root 4096 May 25 18:20 ha.d
drwxr-xr-x 2 root root 4096 May 25 17:37 init.d
-rw-r--r-- 1 root root 1209 May 20 15:55 ipsec.conf
-rw-r--r-- 1 root root  178 Apr 25 13:10 ipsec.secrets
-rw-r--r-- 1 root root   31 Mar 31 00:23 issue
drwxr-xr-x 2 root root 4096 May 25 12:19 network
drwxr-xr-x 2 root root 4096 Apr 25 13:13 pam.d
drwxr-xr-x 2 root root 4096 Apr 25 13:13 ppp
```

```
drwxr-xr-x 2 root root 4096 Apr 27 11:28 samba
drwxr-xr-x 2 root root 4096 Apr 25 18:23 xl2tpd
./includes.chroot/etc/apt:
total 4
-rw-r--r-- 1 root root 545 May 19 13:28 sources.list

./includes.chroot/etc/bind:
total 16
-rw-r--r-- 1 root root  244 May 25 12:29 named.conf.local
-rw-r--r-- 1 root root 1981 May  6 19:16 named.conf.options
-rw-r--r-- 1 root root  940 Apr 25 18:33 naxtaro.com
-rw-r--r-- 1 root root  718 May 20 13:05 naxtaroint.com

./includes.chroot/etc/dhcp:
total 4
-rw-r--r-- 1 root root 721 May 25 17:11 dhcpd.conf

./includes.chroot/etc/ha.d:
total 16
-rw------- 1 root root   39 May  9 19:33 authkeys
-rw-r--r-- 1 root root  213 May 25 18:20 ha.cf
-rw-r--r-- 1 root root  111 May 25 18:20 haresources
drwxr-xr-x 2 root root 4096 May  6 13:29 resource.d

./includes.chroot/etc/ha.d/resource.d:
total 0
lrwxrwxrwx 1 root root 25 May  6 13:27 apt-cacher-ng -> /etc/
init.d/apt-cacher-ng
lrwxrwxrwx 1 root root 17 May  6 13:28 bind9 -> /etc/init.d/
bind9
lrwxrwxrwx 1 root root 14 May  6 13:28 fw -> /etc/init.d/fw
lrwxrwxrwx 1 root root 17 May  6 13:28 ipsec -> /etc/init.d/
ipsec
lrwxrwxrwx 1 root root 27 May  6 13:28 isc-dhcp-server -> /etc/
init.d/isc-dhcp-server
lrwxrwxrwx 1 root root 17 May  6 13:29 mysql -> /etc/init.d/
mysql
lrwxrwxrwx 1 root root 15 May  6 13:29 ntp -> /etc/init.d/ntp
lrwxrwxrwx 1 root root 17 May  6 13:29 samba -> /etc/init.d/
samba
lrwxrwxrwx 1 root root 18 May  6 13:29 squid3 -> /etc/init.d/
squid3
lrwxrwxrwx 1 root root 15 May  6 13:29 ssh -> /etc/init.d/ssh

./includes.chroot/etc/init.d:
total 8
-rwxr-xr-x 1 root root 4928 May 25 17:37 fw

./includes.chroot/etc/network:
total 4
-rw-r--r-- 1 root root 636 May 25 12:19 interfaces.naxtaro

./includes.chroot/etc/pam.d:
total 4
-rw-r--r-- 1 root root 147 Apr 25 13:13 ppp

./includes.chroot/etc/ppp:
```

```
total 8
-rw-r--r-- 1 root root 214 Apr 25 13:12 options.xl2tpd
-rw-r--r-- 1 root root  42 Apr 25 13:13 pap-secrets

./includes.chroot/etc/samba:
total 12
-rw-r--r-- 1 root root 9459 Apr 27 11:28 smb.conf

./includes.chroot/etc/xl2tpd:
total 4
-rw-r--r-- 1 root root 249 Apr 25 18:23 xl2tpd.conf

./includes.installer:
total 16
-rw-r--r-- 1 root root 3866 May 21 02:47 preseed.cfg
drwxr-xr-x 3 root root 4096 Mar 31 05:00 usr

./includes.installer/usr:
total 4
drwxr-xr-x 3 root root 4096 Mar 31 05:00 share

./includes.installer/usr/share:
total 4
drwxr-xr-x 2 root root 4096 Mar 31 05:02 graphics

./includes.installer/usr/share/graphics:
total 24
lrwxrwxrwx 1 root root    15 Mar 31 05:02 logo_debian_dark.png -
> logo_debian.png
-rw-r--r-- 1 root root 20619 Apr 28 16:18 logo_debian.png
lrwxrwxrwx 1 root root    15 Mar 31 05:02
logo_installer_dark.png -> logo_debian.png
lrwxrwxrwx 1 root root    15 Mar 31 05:02 logo_installer.png ->
logo_debian.png

./includes.source:
total 0

./package-lists:
total 8
-rw-r--r-- 1 root root  42 Mar 30 23:55 live.list.chroot
-rw-r--r-- 1 root root 121 May 25 18:10 naxtaro.list.chroot

./packages:
total 0

./packages.binary:
total 0

./packages.chroot:
total 0

./preseed:
total 0

./rootfs:
total 0
```