Installing an Ubuntu PXE Server alongside an Existing WDS Server

Levi Cherry-Allen
T213
Olympic College
10/14/2015

Installing an Ubuntu PXE Server alongside an Existing WDS Server

Levi Cherry-Allen
T213
Olympic College
10/14/2015

Summary

This report is to explain the processes and results of my project of implementing an Ubuntu PXE server into an existing network architecture. The most important goal of my work was to allow a multiple network-boot environment without disrupting the existing infrastructure. This goal was successful, and only required two changes to the WDS server.

Table of Contents

Introduction

The problem for this assignment was threefold. The first was getting the Ubuntu server set up properly, with minimal installs and processes running to preserve performance. The second problem to solve was getting the Ubuntu server to communicate effectively with the Microsoft domain/network. Finally, allowing the technician to easily decide between the two servers without having to make configuration changes every time. Solving this problem has opened up many new options for the lab, including distributing Linux over the network, and adding new functionality later.

Building the Ubuntu Server

This step is first of many in this project, and possibly the easiest. To start download a fresh Ubuntu Server 15.04 .ISO and install it onto a new Hyper V VM. When doing the setup, be sure to use static memory, not dynamic. Using this setting will cause the server to throw errors every few minutes, reducing usability and performance.

Configuring Networking

The next step is to set up networking. To do this, use the command *sudo vi /etc/network/interfaces* and edit as follows.

```
# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet dhcp
dns-nameservers 192.168.33.254 192.168.30.254
```

Type :x and enter to save and exit. Next, restart the server to apply the changes with *sudo /etc/init.d/networking restart*.

Installing Necessary Packages

These packages are all required for our purposes, and to get them all, simply type *sudo apt-get update* then type *sudo apt-get install inetutils-inetd tftpd-hpa nfs-kernel-server ssh-server* (all one line, spaces as shown).

Configuring TFTP Service

To set up this service properly, we will edit two files. The first should be edited with *sudo vi /etc/inetd.conf* as such:

tftp dgram udp wait root /usr/sbin/in.tftpd /usr/sbin/in.tftpd –s /var/lib/tftpboot

There will be other things in the file already, ignore them. Simply add this to the very bottom and use :x then enter to save. The second file that should be changed is *sudo vi /etc/default/tftpd-hpa* to read as:

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
TFTP_ADDRESS="192.168.0.32:69"
TFTP_OPTIONS="--secure" (note: there should be two hyphens --).
RUN_DAEMON="yes"
OPTIONS="-l –s /var/lib/tftpboot"

Now type :x then enter to save. Finally, start the service with *sudo service tftpd-hpa start* and test it with *sudo netstat –lu* which should list tftp in as a listening port.

Obtain, and Configure PXELINUX Boot Files

In order for the PXE server to work, we will need to have a bootfile for it to hand to a computer so that it can then load an ISO and install it. To do this, we will need to download the syslinux package and unzip it. To download it, use the command *cd ~* to change directory to the home directory, then *wget https://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-6.03.tar.gz* to download the file. To unzip it, use *tar –xf syslinux-6.03.tar.gz* now there should be a folder in the home directory called "syslinux-6.03". Inside this folder are the boot files we need. Next, we will put all of the boot files where they need to go, to do this, use the following commands:

sudo mkdir /var/lib/tftpboot
sudo mkdir /var/lib/tftpboot/pxelinux.cfg
sudo mkdir –p /var/lib/tftpboot/Ubuntu/14.04/amd64
sudo cp /usr/lib/syslinux-6.03/efi64/com32/modules/ /var/lib/tftpboot/*
sudo cp /usr/lib/syslinux-6.03/efi64/menu/vesamenu.c32 /var/lib/tftpboot/
sudo cp /usr/lib/syslinux-6.03/bios/core /var/lib/tftpboot/

To change the settings of the menu that displays when booting to the Ubuntu PXE Server, we will create a file *sudo vi /var/lib/tftpboot/pxelinux.cfg/default* to look like:

DEFAULT vesamenu.c32
TIMEOUT 100
PROMPT 0
MENU INCLUDE pxelinux.cfg/pxe.conf

NOESCAPE 1
LABEL Try Ubuntu 14.04 Desktop
MENU LABEL Try Ubuntu 14.04 Desktop
kernel Ubuntu/vmlinuz
append boot=casper netboot=nfs nfsroot=192.168.0.32:/var/lib/tftpboot/Ubuntu/14.04/amd64
initrd=Ubuntu/initrd.lz quiet splash
ENDTEXT
LABEL Install Ubuntu 14.04 Desktop
MENU LABEL Install Ubuntu 14.04 Desktop
kernel Ubuntu/vmlinuz
append boot=casper automatic-ubiquity netboot=nfs nfsroot=192.168.0.32
:/var/lib/tftpboot/Ubuntu/14.04/amd64 (continued from above line)
initrd=Ubuntu/initrd.lz quiet splash
ENDTEXT

To save, type :x then enter. Next, to adjust how the menu looks, another file must be edited, by using the command *sudo vi /var/lib/tftpboot/pxelinuz.cfg/pxe.conf* to edit as below:

MENU TITLE Ubuntu PXE Server
NOESCAPE 1
ALLOWOPTIONS 1
PROMPT 0
MENU WIDTH 80
MENU ROWS 14
MENU TABMSGROW 24
MENU MARGIN 10
MENU COLOR border 30;40 #ffffffff #00000000 std

To save, type :x then enter.

Setting Up Boot Images

Now that the server has been mostly set up, it needs media to boot, this is in the form of an extracted .ISO file. To do this, enter the following commands. *sudo /mnt* to change directory, then *sudo wget http://releases.ubuntu.com/14.04/ubuntu-14.04.3-desktop-amd64.iso* to download it. Next we will extract the .ISO with the following commands:

sudo mount –o loop /mnt/ubuntu-14.04.3-desktop-amd64.iso /media/
sudo cp –r /media/ /var/lib/tftpboot/Ubuntu/14.04/amd64*
sudo cp –r /media/.disk /var/lib/tftpboot/Ubuntu/14.04/amd64
sudo cp /media/casper/initrd.lz /media/casper/vmlinuz /var/lib/tftpboot/Ubuntu

Configure NFS Server to Export ISO Contents

Next, the server must be set up to properly export the contents of the .ISO to clients. This is done by using the command *sudo vi /etc/exports* and add the following line to the file: */var/lib/tftpboot/Ubuntu/14.04/amd64 *(ro,async,no_root_squash,no_subtree_check)* now save and exit the file by typing :x then enter. Now apply the settings and start the server with the commands *sudo exportfs –a* and *sudo /etc/init.d/nfs-kernel-server start* now the server should be ready to accept clients.

Setting Up Server Boot Environment

Although the server is working, upon a restart there will be problems because the tftpd-hpa service is not set to start. This can be done with a command, but leads to problems, because inetutil.inetd will use port 69 and prevent tftpd-hpa from properly binding. To solve this, disable inetutil.inetd from starting automatically with *sudo update-rc.d inetutils-inetd disable* and *sudo update-rc.d tftpd-hpa disable* (this line might not actually be needed) and edit the startup environment by editing rclocal with *sudo vi /etc/rc.local* and add the following lines ABOVE "exit 0":

sudo service start tftp-hpa start
sudo service inetutils-inetd start

Type :x and enter to save. The server should be ready and working, even if it gets restarted now.

Joining the Ubuntu Server to the Microsoft Domain

In order to use the Windows Domain, some settings must be applied to the Ubuntu server. This allows us to use the Windows DNS, along with making the DHCP happier when we have to modify it. First we need to download PowerBroker Identity Services (PBIS), With the command *sudo wget http://download.beyondtrust.com/PBISO/8.3/pbis-open-8.3.0.3287.linux.x86_64.deb.sh* (one line). Next, change the permissions of the file with *sudo chmod a+x pbis-open** and run it with *sudo ./pbis-open-8.3.0.3287.linux.x86_64.deb.sh* and answer "yes" when it asks about legacy links and "yes" when it asks to install now. Finally, join the Microsoft Domain with the command:
sudo domainjoin-cli join 'domain' 'domain admin account@domain' Substitute the contents of quoted material with the relevant information, and remove the quotes. It will prompt for a password, input the password of the admin account, then when it is complete, restart the Ubuntu server with *sudo reboot* and all of the configuration on the Ubuntu server is complete.

Setting up the Microsoft Network

The two things that need to be changed on the Microsoft network to ensure reliability and functionality are the DNS and DHCP server.

Configure DHCP Settings

To setup the DHCP settings, log into the Windows DHCP server. The first thing we will need to do is create a DHCP reservation for the Ubuntu Server. Locate the entry

“ubuntupxe.domain”, right click on it, and press “Add to Reservation”. NOTE: The IP Address of this machine may not be 192.168.0.32, and if so, **the ip settings in the Ubuntu configuration files will need to be changed accordingly.** Next, we will have to change some server options. Still in the DHCP Manager, select IPv4 and open the “Server Options” menu, right click, and use “Configure Options” to add options 066 and 067. 066 Should point to the WDS server (in my case 192.168.0.241), and 067 should read “\boot\x64\wdsnbp.com”.

Configure DNS Settings

The DNS settings are not strictly necessary, but it allows us to call the server by a name instead of the IP address. To do this, open the DNS Manager, and add a record in the forward lookup zone called “UBUNTUPXE” and link it accordingly to the IP address of the server.

Setting up the Microsoft WDS Server

With the Ubuntu server now set up, the only thing that still needs to be completed is changing options in the WDS server to give the client a menu to choose which server to use.

Downloading Syslinux Boot Files

While on the WDS server, download Syslinux from: *https://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-6.03.zip* and unzip it to a temporary location. Next, open the folder and copy *core/pxelinux.0*, *modules/pxechain.com*, and *com32/menu/menu.c32* to the WDS remote install folder (for me, *D:\RemoteInstall\Boot\x64*). Now, navigate to the remote install folder and make copies (keep the old files) of *pxeboot.n12*, and *abortpxe.com* then rename the copies *pxeboot.0* and *abortpxe.0* respectively.

Configuring Syslinux Boot Environment

First, create a new directory in the *\RemoteInstall\Boot\x64* folder called *pxelinux.cfg* (yes, a folder with a file extension). Next, create a new file in this directory, *\RemoteInstall\Boot\x64\pxelinux.cfg\default* (note, no .txt file extension) and edit it as follows:

```
DEFAULT menu.c32
MENU TITLE WDS PXE Server

LABEL wds
MENU DEFAULT
MENU LABEL Windows Deployment Services
KERNEL pxeboot.0

LABEL abort
MENU LABEL Abort PXE
KERNEL abortpxe.0

LABEL linuxpxe
```

MENU LABEL Linux PXE Server
KERNEL pxechain.com
APPEND 192.168.0.32::pxelinux.0
 # IP Address Above is Ubuntu PXE Server

Finally, apply these settings by running the following commands in an elevated command prompt:

wdsutil /set-server /bootprogram:boot\x64\pxelinux.0 /architecture:x64
wdsutil /set-server /N12bootprogram:boot\x64\pxelinux.0 /architecture:x64

Clean Up Everything

Now that everything is working correctly (assuming that tests have been done, and Linux is successfully deploying, we need to clean some things up to ensure that the server is as clean as possible, and easier for the next person to maintain. First of all, cleanup everything in the Ubuntu home directory with the command *sudo rm –r ~/** (be careful there is nothing important in there first). Next, unmount the Ubuntu .ISO with *sudo umount /mnt* and remove it from the mount folder with the *rm /mnt/** command. Finally, clean up the WDS server by removing the syslinux.zip we downloaded, as well as the temporary location that it was unzipped to.

Conclusions

This project allows a WDS server, and an Ubuntu PXE server to coexist on the same network, peacefully. The client first connects to the DHCP server with a PXE request, and the DHCP redirects the client to the WDS server. This server loads up a small *pxelinux.0* kernel, which displays a menu and allows the user to choose to load a new bootfile from either the WDS (*pxeboot.0*), or the Linux PXE Server (*pxechain.com* loading into *pxelinux.0* on the Ubuntu PXE). This method of chain booting basically allows us to point the client at a single server, then let them choose what ACTUAL server they want to use.

References

Jethva, H. (2015, September 15). Configure PXE Server In Ubuntu 14.04. Retrieved September 21, 2015, from https://www.maketecheasier.com/configure-pxe-server-ubuntu/

Gray, E. (2011, June 27). Peaceful Coexistence: WDS and Linux PXE Servers. Retrieved October 1, 2015.