FreeBoo: an Open Architecture for Network Dual Boot

April 1st, 2009 by Cristina Barrado and Sebastian Galiano in SysAdmin

Using FreeBoo, you can restore and boot different operating systems across a network and replace proprietary solutions such as Rembo.

Administrating large installations of computer desktops requires many tedious system reparations due to software updates, hardware fixes, user mistakes and viruses. To reduce costs, some enterprises adopt restrictive IT politics. But, if your business cannot afford a highly secure and restrictive environment, and you want to provide your many users with dual-boot capacity, desktop administration privileges and the possibility to execute a large amount of different software, you probably are using Rembo.

This article presents FreeBoo, an open architecture that provides you with a dual-boot system of secure desktop images. FreeBoo is based on network boot, provides image restoration and allows hot boot. With FreeBoo, any malicious software installation done on a desktop by a previous user can be overwritten seamlessly.

Image Restoring and Dual Boot

Many IT departments' efforts are dedicated to the time-consuming task of repairing end-user desktops. For this task, most IT systems use open-source imaging systems that exist today, such as SystemImager, partimage, FileZilla, clonezilla, Frisbee, rsync, rdiff-backup, ADIOS and so on, or their commercial equivalents, including Norton Ghost, Active, True Image and Image. All these tools create a compressed image of a client's hard drive data and save it in case a future data recovery is necessary. An image is the complete copy of a filesystem, and it usually is stored on a backup server. When image changes are small, incremental backup is used to improve performance.

Imaging systems use well-known IETF protocols to transfer data from client to server or vice versa. They also include many functionalities for image management, with easy-to-use GUIs. IT departments in charge of large installations also use them to clone OS images onto several identical computers and to update systems with new patches.

In general, this software requires a high level of expertise, works basically on-demand and runs with a client program. This last feature is very important, because it assumes that the client computer is executing with specific conditions. Typically, this means the client always executes the same operating system.

At our university, computer labs can boot either MS Windows or Linux operating systems, and students select the desired partition using Rembo. Other PC-compatible dual-boot options include Norton BootMagic, OSL2000 or the MSTBOOT commercial systems and the GRUB open-source software solution. But, none of these tools can dual boot from the network.

Rembo is the only existing tool that provides the option of restoring any of the computer's saved images. And, very important, once the image is recovered, the computer boots it directly. Rembo is a commercial evolution of the open-source BP Batch Project, recently integrated into the IBM Tivoli suite. Rembo introduces local disk caches for fast image restoration, is able to use multicast messages and can be programmed using the Rembo-C scripting language.

FreeBoo Architecture

FreeBoo is proposed as an open-source, alternative solution to Rembo. Instead of open-source software, FreeBoo is an architecture built from many existing open-source programs. In fact, the number of new lines of code is insignificant. We have written only eight simple scripts and have used the urldecoder script authored by Heiner Steven. Figure 1 shows the global picture. The open protocols used include TFTP, DHCP, HTTP and NFS. And, the open-source code, running either on the server or client includes PXELinux, rsync, partimage, Apache, Netcat, links2 and gensplash. In Figure 1, you can see which program is executed at each computer site (client or server).

Figure 1. Global View of the FreeBoo Components

To illustrate the use of FreeBoo, let's assume the following scenario: ten or more desktops

connected by LAN with a remote, non-accessible server room, using a PXE-compliant NIC. The desktops boot in three phases, as shown in Figure 2. The first phase is similar to a thin-client boot. In the second phase, the user selects a boot option, and the necessary image data is sent to the client. Finally, in the third phase, the client computer boots the user-selected OS.

Figure 2. FreeBoo Phases

Thin-Client Boot Phase

To build a FreeBoo system, you first need to configure your client's BIOS to boot from the network and to dedicate a server for PXE. You must start the DHCP and TFTP dæmons on your server and use the Pxelinux.0 file as the PXE primary bootstrap.

The main consideration regarding your thin-client configuration is the actual image properties. Because it is only a temporary image, let's configure a small embedded Gentoo Linux with generic drivers (less than 6Mb). We will add the FUSE drivers, the kexec system call introduced in the 2.6 kernel and NFS, rsync and partimage client utilities. FUSE is required, because we need write access to the NTFS partition during image restoration. kexec is used for the hot boot. The thin client uses an NFS root filesystem to avoid the initial RAM filesystem transfer done by most thin-client solutions. The NFS option is slower for individual access but faster for an initial deployment (more on using rsync and partimage in the next section).

To get a nice user interface, we also add the framebuffer driver, Gensplash (Gentoo's bootsplash software) and the Links2 browser compiled with direct framebuffer support. The Links2 text browser and Gensplash text images for framebuffer let you avoid the cost of the X Window System and its configuration problems, while achieving some graphical capacities.

Image Selection and Restoration Phase

The second phase consists of presenting a boot menu to the end user and (if required) executing the image restoration. This FreeBoo phase starts when the client PC is running the Linux thin client. You should configure it to execute the Links2 text Web browser initially. Simply add this init.d file to your thin-client filesystem (located in the server's drive and accessed via NFS):

```
#/etc/init.d/freeboo
depends () {
  after gpm
  after local
  depends local
}
start() {
  /scripts/freeboostart.sh
  eend 0
}
```

The freeboostart.sh script executes the Web browser:

```
#!/bin/sh
# Part of the /scripts/freeboostart.sh file
links2 -g server_IP
# ...
```

The Apache Web server responds with an HTML file that contains the end-user menu. Figure 3 shows the available boot options. The user can choose to boot a given OS directly or to restore its saved copy. If the user selects a direct boot, FreeBoo skips directly to phase three. Otherwise, FreeBoo proceeds first to transfer the OS image to the client.

Figure 3. Boot Menu

The menu provides two image restoring policies: complete (or full) restoring and partial (or fast) restoring. Complete restoring transfers all of the image to the client computer using partimage. Partial restoring transfers only the modified data using rsync. Depending on the differences with the current image, one solution is faster than the other. We have found that for partial restoring, the checksum computation was especially time consuming for large files. But, both solutions achieve the same goal—to deploy a clean and secure OS image on the client.

In addition to configuring Apache and setting the boot menu page as the user's initial page, you also need to start the rsyncd and the partimaged dæmons on the server. rsyncd, for partial restoring, uses virtual names for the OS images. Its configuration file (/etc/rsyncd.conf) assigns each virtual name to a path in the filesystem. You should create a disk partition for each OS image, because rsync is filesystem-sensitive. Specifically, you need an NTFS filesystem and the FUSE ntfs-3g mount command for writing the MS Windows image on the client. We recommend excluding the big and unneeded MS Windows virtual memory file (pagefile.sys) and the hibernate file in the rsync recovery command (flag --exclude-from). We also recommend deleting any user-created files on the client (flag --delete). Additionally, add the flags -a (maintain permissions, ownership and timestamps), -r (recursive) and -v (verbose). Below is the code for the syncroLinux.sh and syncroWindows.sh scripts. They differ only in the mount command and image and partition names:

```
#!/bin/sh
# /script/syncroWindows.sh and /script/syncroLinux.sh
# FreeBoo scripts for partial image restoring
#
# INSERT the correct mount command
#   for Linux:      mount /dev/hda2 /mnt/linux
#   for MS-Windows: ntfs-3g /dev/hda1 /mnt/windows -o force

ntfs-3g /dev/hda1 /mnt/windows -o force

# INSERT the correct rsync command
#   rsync -avr --delete --exclude-from=.rsync/exclude \
#         SERVER_IP::SYNC_NAME DEST_FOLDER

rsync -avr --delete --exclude-from=.rsync/exclude \
      192.168.1.1::windows /mnt/windows

# ... continues with kexec commands ...
```

Full Image Restoring

For full image restoration, we use partimage. partimage is much simpler than rsync, because it is not filesystem-dependent. You simply need to create a directory on the server to store the partimage chunk files. Chunks are just data that can be stored in any ext3 filesystem. Because the client gives the full path to the image chunks, you don't need any special configuration for partimaged. The flags to add are -f3 (quit when finished), -b (batch mode) and, for performance issues, -z# (level of compression). Below is the code that restores full images (the restoreLinux.sh and restoreWindows.sh scripts):

```
#!/bin/sh
# /scripts/restoreLinux.sh and /scripts/restoreWindows.sh
# FreeBoo scripts for full image restoring
#
# INSERT the correct partimage command
#   partimage -f3 -b -s SERVER_IP restore DEST_DEVICE IMAGE
partimage -f3 -b -s 192.168.1.1 \
          restore /dev/hda2 /root/fileImage/linux

# ...continues with kexec commands ...
```

Hot Boot Phase

The third, and most challenging, FreeBoo phase is the image hot boot. When we started working on FreeBoo, our first idea was to enter into the Linux kernel code to write the hot boot function. But while examining the code, we discovered the kexec system call and its related shell commands. The following two scripts are able to hot boot an OS image installed on a local drive.

Linux hot boot script:

```
#!/bin/sh
# /scripts/startLinux.sh
# FreeBoo script for Linux hot boot
#
mount /dev/hda2 /mnt/images/linux
kexec -l /mnt/images/linux/vmlinuz \
      --append="root=/dev/hda2 ro quiet splash" \
      --initrd="/mnt/images/linux/boot/initrd.img-2.6.15-23-386"

kexec -e
```

MS Windows hot boot script:

```
#!/bin/sh
# /scripts/startWindows.sh
# FreeBoo script for Windows hot boot
#
ntfs-3g /dev/hda1 /mnt/images/windows -o nonempty
kexec -l /mnt/images/windows/grub.exe
kexec -e
```

Both scripts first mount the partition of the OS image, then they execute kexec -l, and finally, they execute the new OS kernel with kexec -e. The OS kernel must be an ELF executable. For Linux, this is the kernel file directly, but for MS Windows, we use the GRUB bootloader (you should replace the NTLR default bootloader with the GRUB bootloader before saving the client MS Windows image on the server). Finally, you need a GRUB bootloader, such as Grub4dos, with built-in BIOS disk emulation and an ATAPI CD-ROM driver. This is needed because the thin-client execution overwrites the BIOS and most MS Windows versions still rely on some BIOS interrupts for video, timers and disk-related hardware I/O.

Script Execution

The final consideration to get FreeBoo working is in respect to the execution of the script files. We have one script file for each of the six alternatives in the user's menu. These files are located on the server, but we need them to execute on the client. In the menu's HTML code, the links to the scripts are localhost references—for example, <a href="http://localhost/scritps/startLinux.sh">, so the Web browser will try to connect to a local Web server. But, we have not installed any Web server on the client image, because we wanted a light and fast kernel. Instead, we have included two additional scripts in FreeBoo to provide a local Web server. Both scripts are initiated by /etc/init.d/freeboo. Below are the complete freeboostart.sh and the new mini_webapp_s.sh scripts.

Initial freeboostart.sh script:

```
#!/bin/sh
# /scripts/freeboostart.sh
# FreeBoo script that starts a very simple http
# server with script execution capacity
# in parallel with a web browser
#
mini_webapp_s.sh &
links2 -g YOUR_SERVER_IP
```

The mini_webapp_s.sh script:

```
#!/bin/sh
# /scripts/mini_webapp_s.sh
# FreeBoo script that parsers the incoming browser
# request, gets the script path name and
# executes it locally
#
torun=`nc -l -p 80 -s 127.0.0.1 | \
         awk '/HTTP/{print $2; exit}' | \
         urldecoder.sh`
($torun)
```

The first script starts the links2 Web browser mentioned previously. But, it also starts the mini_webapp_s.sh script in parallel to act as a very simple Web server with application execution capacities. This second script executes the output of a pipeline command composed of netcat, awk and urldecoder, which extracts the filename of the script to execute.

Netcat (nc) is a very simple command. Like the traditional cat command, netcat simply copies data from an input stream to an output stream; the only difference is that these streams can be network data. The -l flag (listen mode) specifies that netcat's input comes from the network. The -p 80 and -s 127.0.0.1 options indicate that the input will come on port 80 (the HTTP default port) from IP address 127.0.0.1 (localhost). Netcat's function is to redirect any HTTP request, like the one below, to the awk filter:

```
GET /scripts/syncroWindows.sh HTTP/1.1
Host: localhost
User-Agent: ...etc...
```

The awk command extracts the script filename found on the HTTP GET line and passes it to the urldecoder.sh script through a second pipe. urldecoder.sh is a well-known script used to convert a URL with special characters, such as blank spaces, to a valid filename. The parser pipeline finishes when the HTTP of the GET line is found. Then, the variable $torun is set with its output and immediately executed. In the example above, the user has selected the MS Windows fast restore, and the HTTP request contains the /scripts/syncroWindows.sh filename. The pipeline extracts this name, and the next line executes it on the client.

The links2 browser, which is executing in parallel, is waiting for the HTTP response from our local Web server. Because we don't need any more interaction with the user, instead of sending back a response, we have included a line to kill the links2 process in each of the six menu scripts:

```
# first line of {start | syncro | restore}{Linux | Windows}.sh scripts
killall links2
```

Conclusion

The BP Batch Project, started at the Geneve University by Marc V. Stuckelberg and David Clerc, became a popular open implementation of the thin client. This approach evolved into the commercial Rembo suite, which is used in many labs with a significant licensing cost. FreeBoo uses a combination of existing open-source technologies, including BP Batch, to provide the main features of Rembo.

The hardware requirements for installing FreeBoo are just a dedicated server connected to client desktops by a LAN. Desktops need to have only boot-on-LAN capacities and local disk drives. All the software used is open source.

Future extensions of FreeBoo include the use of this technology for server software deployment; the development of a Web-based interface for easy administration of images, including database management; evaluation of the performance of the OS restoration process to improve it and to select the best option automatically (instead of having the user decide between the fast or full options); the insertion of multicast image recovery; and finally, the use of Wake-on-LAN capabilities to deploy secure images to desktops at preprogrammed times.

FreeBoo is only the initial step in building an open-source boot environment for system administrators that allows you to fix, deploy and execute OS images on large installations of desktops.

The scripts and other files related to FreeBoo can be found at ftp.linuxjournal.com/pub/lj/listings/issue180/10203.tgz.

Cristina Barrado is an Assistant Professor at the Technical University of Catalonia, in Castelldefels, Spain. She has been teaching operating systems since 1990

and has advised a large number of master theses on Linux and Linux development. Her PhD was focused on automatic extraction of low-level parallelism in loops at compile time. Currently, she belongs to the Icarus Research Group, whose target is research on avionics systems for Unmanned Aerial Systems.

Sebastian Galiano is a Telematic Engineer with extensive OS and network services knowledge. Currently, he is working at UPCnet as project engineer in the Internet and middleware area.

Comment viewing options

Threaded list - expanded | Date - newest first | 50 comments per page | Save settings

Select your preferred way to display the comments and click "Save settings" to activate your changes.

new
Join forces to build a Rembo/Mokafive replica
On March 14th, 2009 HenrikBach says:

If there are some guys out there interested in joining forces for building a Rembo/Mokafive replica, count me in. We could setup a site on one of those floss sites...

Thoughts?

reply

new
Hi I'm Sebastian Galiano
On March 16th, 2009 sebbasman says:

Hi

I'm Sebastian Galiano (One of Freeboo's authors), If you need any help you can count on me. About the tar.gz you are requesting, maybe I can help on that too. Do you need any specific file?

reply

new

FreeBoo project

On March 19th, 2009 HenrikBach says:

Hi Sebastian!

That sounds great, that I can count on you. First, Now I'm able to download the tar.gz. I'll return back later, or you can write to me at bach.henrik at gmail dot com.

reply

new

I cannot download the files related to this article

On March 14th, 2009 HenrikBach says:

I cannot download the files related to this article: ftp.linuxjournal.com/pub/lj/listings/issue180/10203.tgz.

It seems that the folder containing the compressed archive doesn't exists.

Can you have a look at that?

reply

new

ftp site

On March 11th, 2009 georgecorondan says:

liked your system admin issue 180 but when i tried to get the code for:
FreeBoo: an Open Architecture for Network Dual Boot
there was nothing on your ftp site! do you guys need a sys admin :-) ?

reply

Post new comment

Please note that comments may not appear immediately, so there is no need to repost your comment.

Your name:
hjmangalam

Subject:

Comment: *

- Allowed HTML tags: <a>   <cite> <code> <pre>   <dl> <dt> <dd> <i> <b>
- Lines and paragraphs break automatically.

More information about formatting options

Preview comment