

Putting OSX in an Open Access Lab (or “The Joy of X”)

David L. R. Houston
University of Vermont
Computing & Information Technology
238 Waterman Building
011 · 802 · 656 · 2013
David.Houston@uvm.edu

ABSTRACT

This paper discusses the challenges of putting Apple Macintosh OSX into open access and computer lab environments.

Categories and Subject Descriptors

K.6.4 [Management of Computing and Information Systems]: System Management – *management audit, quality assurance.*

General Terms

Performance, Design, Reliability, Security, Human Factors, Standardization, Labs, Integration

Keywords

Open Access, Lab, OSX, Macintosh, Workstation, Maintenance, Software Distribution, Configuration, Security, Imaging.

1. INTRODUCTION

Macintosh OSX has been around for a few years, and lab managers are still struggling to wrestle it into a lab environment. Many lab managers are new to OSX, may be new to Macintosh and may be even newer to UNIX. Learning how to configure, image and deploy this new breed of Mac OS is a daunting challenge, and often a process of experimentation. It doesn't have to be this difficult.

2. WHAT IS OSX?

OS X is Apple Computer's latest operating system. It is a major departure from the more traditional Macintosh operating system, despite the fact that the roots of the system predate the original Mac OS. The changes bring a fresh and vital system to computing.

As a candidate for use in an open access computing lab in a college or university setting, OSX offers a lot of promise. With this promise come a raft of challenges, all of which require care and feeding. It may seem daunting at this stage – you are, after all, mapping out a plan to deploy many UNIX workstations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGUCCS'03, September 21-24, 2003, San Antonio, Texas, USA.
Copyright 2003 ACM 1-58113-665-X/03/0009...\$5.00.

Nevertheless, it is doable. You too can wrestle with alligators and know the true “Joy of X”.

Fundamentally, OSX is a *UNIX* system. For those of you well versed in System 9 and earlier, this will be the most significant change. You might notice the most radical change in the idea of a *command line interface*, something that was entirely absent in all previous versions of the Macintosh OS. This is probably the one distinguishing characteristic of OSX when contrasted with earlier Macintosh operating systems. The command line interface or *CLI* may seem daunting. If you cut your teeth on the earlier Mac OS such as OS9, using a CLI is going to take some getting used to. Don't be dismayed. It is a lot easier than you might think.

3. THE TOOLKIT

As with all things computing, we need a toolkit that makes the job not just easier, but in this case, doable! This section inventories the tools you will need to get the job done.

3.1 One Machine for Master, One for Cloning Tests

I am a strong proponent of having the machine you are working on available as part of the toolkit. The reality of budgets may preclude this as an option. Clearly, *not* having a machine to work on, to “break”, or to experiment with, is going to limit your options. If you face a restriction here, I would probably suggest you seriously look at setting up an older machine *first* to really twiddle and break.

If you can, get a second, identical piece of hardware to work with. This second machine is the one you will really “burn” several times. It is where all your practice cloning takes place. Dolly the sheep never had it so good!

One important axiom has been a cornerstone of my work on all platforms: build your master image in layers, beginning with the most basic and best understood. Add more as you gain confidence.

3.2 Carbon Copy Cloner

Carbon Copy Cloner is the brainchild of Mike Bombich (www.bombich.com). It is an AppleScript Studio “interface” to existing command line tools (surprise!): *asr* and *ditto*. Carbon Copy Cloner or *CCC* offers a means of backing up an OSX disk. In addition to making backups, CCC is also a great tool for the production of your master image file. It takes a complex set of

command line options and forces the resulting image file to be fully *asr* “ready” for use by NetRestore and later cloning.

3.3 NetRestore

NetRestore is also from Bombich. It is a marvelous little utility designed to allow the restoration of a complete hard drive image. The source image can be on a local partition, on a FireWire drive, on a CD (if it fits, of course!) or on the network somewhere.

Use is very simple and quick. A FireWire drive used as the boot device with a 2-gigabyte image file on that same FireWire drive can restore to the local hard disk in about 10 minutes. In addition, it is possible to do some critical *post-processing* using shell scripts.

3.4 FireWire Drive

The changes in storage recently are nothing short of astonishing, and adding a FireWire drive to your stable of goodies is a really good idea. It is faster, very versatile and gives you a lot of flexibility in how you do this job. I have used a LaCie PocketDrive with excellent results, but almost any FireWire drive should work fine for new systems.

Your toolkit is one of the most important things you can develop as you wrestle with the OSX Alligator.

4. SECURITY

Security is practically the centerpiece of how the process of imaging and cloning takes place. There is a lot here that is very new and very different for experienced Mac OS9 users and administrators.

Why is security important? It is easy to set up an *Apache web server*, the most commonly used web server in the world today, on an OSX box. I can think of a few things I would rather not have to deal with on an unmanaged web server. It is easy to configure *ssh* and allow anyone in. It is easy to set up packet sniffers that discretely examine every packet, conveniently emailing the results. Are these things you want to leave open to chance? Security has to be the starting point for a good installation.

4.1 Open Firmware

Introduced in recent models, *Open firmware* is actually not new with OSX. Earlier machines also had this feature, and it provides some useful features. With OSX, though, it provides some big security holes, ones that you must take care to protect.

Open firmware is a way for a machine to access certain kinds of parameters at boot time. These settings are stored in non-volatile RAM space. It is similar to the older *parameter ram*.

4.2 Single User Mode

Single user mode is a UNIX feature designed to allow a system administrator access to an ailing machine. When a UNIX box boots into single-user mode, it is generally very limited in what kinds of services it can offer, and it is not enabled for multiple users.

For the OSX system, this same rule applies. Once booted into single user mode, the *root* account is automatically logged in and does not require a password. After this, it is a simple process to check the disk and mount the entire file system as *read-write* and go to work.

4.3 Root

You probably noticed the term *superuser* or *root* in the preceding discussions. These may be new to Macintosh users experienced in previous OS versions. The *root* user is special. This particular user can do anything to a system. As a result, you must use great care. An inadvertent *rm* could easily render your wonderful master image into an attractive bookend!

By default, OSX ships with the root account disabled. This means that should you encounter a situation that requires root access (and there not a lot of these, but they do crop up), you *might* have to enable it. Rather than do this, though, I recommend use of the *sudo* facility, which allows specified users to do things that root must do without the risks inherent in having the root account enabled.

4.4 Local Accounts

As part of the larger security audit, I have moved away from having any *local accounts* at all on OSX client machines. You create local accounts when you first install OSX. Note that for a new installation, the default account type for the first user created in the installation and setup process is that of an *administrative user*. Since administrative users are automatically allowed to use the *sudo* facility, you need to be sure that you manage this carefully.

Instead of local accounts, I elect to have only non-local accounts where users authenticate over the network. To get around the problem of how to do installations as administrator, I configure two special password files, one with passwords in place and the other with all accounts flagged as being not able to log in (use of the * in the *passwd* file in place of the actual password hash does this).

4.5 Why Classic Mode Should Go Away

Classic mode is the add-on to OSX that makes it possible to run older “legacy” applications under OSX. There are, as of this writing, a number of “must-have” applications that are not OSX native and so require Classic mode to run. If you are in a position where you *must* offer such applications, then you are going to have some extra work. Since I have decided not to offer Classic at all, I can’t cover the issues in how to make such a setup secure. There are some excellent resources on the net.

Even with Classic secured, there are some potentially serious security issues at stake: with Classic installed, it is possible to boot into OS9 and then destroy the entire OSX installation. A user can run applications like *FWSucker*, revealing the Open Firmware password. Under OS9, any user can find the password file for OSX and, if you still have local accounts, crack it. In short, Classic poses a measurable security risk and should only be used if it essential.

5. CONFIGURATION

Now we can begin to look at the nitty-gritty details involved in configuration. This section is broken down into specific categories, and begins with the essential element of security.

5.1 Open Firmware

As noted earlier, the very first step is to secure Open Firmware. Do this by booting the machine and holding down the **⌘ OPTION O F** keys. The command line interface appears.

Your first step is to set a password, a good one that you will not forget. The command to do this is simply *password*. If you do forget the password, you must do some fancy footwork to remove that password before you can continue.

After you set the password, you need to set the *security mode*. This is the level of access granted depending on password requirements and how secure you want to make it. Note that once the *command* level of security is set, you cannot boot into single user mode, nor can you boot from a CD until you remove the command level security mode. The command to do this is *setenv security-mode* and the level of security we want here is *command*.

With Open firmware secured as described here, you do not have to worry about single user access.

5.2 Authentication

We have chosen to use LDAP to authenticate users. OSX offers several options, including Active Directory and OSX server. You can find details for those methods at Apple.

Setting up LDAP is quite straightforward. There is one important caveat: you should either have control of the LDAP server yourself or have a good working relationship with those that do! You might need access to logs, or there may be a need to change or modify some of the data or the schemas used to set up the server.

The first step to setting up LDAP uses the OSX utility *Directory Access*. This GUI interface facilitates easy configuration of much of the authentication infrastructure you need for OSX. Note that this part of the paper *only* deals with LDAP v3. This version allows fully encrypted passwords (V2 did not) and SSL support, both essential for a secure installation.

We'll perform the following steps to setup LDAP authentication:

- Configure Directory Access on the local machine
- Create the dummy account
- Add the certificate to the local machine
- Edit the *ldap.conf* file to make the local system aware of the certificates
- Configure Authentication on the client

Start Directory Access, and examine the window that appears.

Under the *Services* tab, you'll see a list of the many services you can use. My configurations turn off *all* services except for LDAPv3. You must first configure the LDAPv3 service (make sure it is checked on). Enter a *search base* and then complete the entry of the *Record Types and Attributes*, and finally set up the

Connection settings. You may wish to enable SSL after you have successfully logged in to the client machine at least once. Note that you *must provide* both the *UID* and the *HomeDirectory* attribute in order to have a functional client with LDAP.

When the LDAP configuration is complete, you need to add this object to the list of things that the client machine can consult at login time. In addition, you must place the certificate used by the LDAP server in the correct location, */System/Library/Openssl/certs/*. Finally, edit the conf file, */etc/openldap/openldap.conf*, to let the client know the LDAP essentials.

5.3 Configuring What Your User Sees

This is a matter of logging in as the “generic” user and configuring the “look and feel” just the way you want it to be. At this point, you do not have to worry about setting it up to prevent the users from changing things – that gets resolved when we use *loginhook* later. Make sure you run every single application at least once, or you may have users trying to enter registration data. You probably want to be sure to set the screen saver to kick in after a few minutes, and be sure to have the password requested when it “wakes up” from the screen saver. I chose to set the sleep pattern to allow for full display sleep, but set no sleep for the CPU, which makes use of the *cron* facility possible (see below).

5.4 Printing

Printing is not as difficult as you might think. In essence, you configure your printers as the generic user, test them to ensure that they work correctly, and when you are done with everything – just before you prepare the master image – you simply change the permissions on both the Print Center application and some special files found in the */etc/* directory.

5.5 Login/Logout Hook

Apple provides a handy mechanism that lets us perform certain tasks at login or logout. To get this to work, we *very carefully* edit a mission critical file – *ttys*. This file can determine if the machine can start at all, so, *make a copy first*. Then edit it and save it. In brief, what you are doing here is “pointing” to a special *shell script* that can run at login or logout, and this is where we call the process that cleans up the user home directory each time. You can find the details for what this line must look like in a number of places on the net (google *loginhook* to find any number of pages).

5.6 Cron Jobs

The UNIX *cron* facility makes it possible to “schedule” things to happen at certain times. We could detail many possible tasks to do for OSX clients, but the one I think is most useful is to have the machine power off at a set time each day or night. You don't have to do this, of course, and will not want to if you run a 24 hour lab, but if you like the convenience of having your OSX machines reliably turn off, it is a real help. *Cron* is easy to use and set up. You must do so as root for most of the useful system processes (like the automatic shutdown). If you read the *man* page on the OSX client for *cron*, you will find a good set of instructions.

5.7 Duplicating the /Users/customer/ folder

At this point, I am not using any sort of full refresh system for the OSX clients as I have done in the past. I plan to do this at some point, but have discovered that, unlike OS9, since users cannot do much to the machines, it really is not needed as often. What I do employ is a simple scripting trick to make sure that every user starts with a fresh, consistent interface, settings and a clean home directory. This is easily accomplished.

After you have completed *all* of your configuration issues, log in as an administrative user. Now, open the Terminal application, become root (use *sudo tcsh* to do this) and then make a special spare copy of the generic user home directory *before* anyone has logged in or done anything to this master machine. I keep mine in */Users/admin/Restore/*. To make use of this backup, I have a simple *shell script* that uses *ditto*, the Apple supplied copy tool, and several standard UNIX utilities to first move any user added *files* to a Lost and Found directory (which you have to create), and then restore the entire */Users/customer/* directory from the hidden spare. This method has caught all of the changes that a generic user has been able to make and is a simple and reliable way to ensure consistency.

5.8 Tweaking the User Interface

Depending upon your tastes and need, you may want to do things like customize the Apple Menu (I remove the *Sleep* choice) and replace the Login screen. Both of these things are well detailed on the web. Replacing the login screen is a bit of a two-step process as one must be root to do this.

5.9 Software Updates

Once you make the local accounts inactive, you may realize that the software updates that you are accustomed to from Apple are inaccessible (don't forget to disable automatic software updates for the generic user!). Apple provides a nice alternative to this as a command line call: *softwareupdate*. Again, one must be at the administrative level or above to make full use of it, but it does mean that you can, if you wish, carefully enable remote access to your client machines using *ssh* and at any given time, simply *ssh* in as an administrative or *sudo* allowed user and run software updates.

5.10 Locking Things down

This is actually very simple. For the most part, a non-administrative user on an OSX box does not have many privileges anyway, so there is not a whole lot to lock down. My approach has been to log in as a *sudo* allowed user, open the Terminal application, and use the UNIX *chmod* command to change permissions. A detailed description of permissions is beyond the scope of this paper, but is the same process involved with the setting of permissions for web pages. I have set many of the applications in the */Applications/Utilities/* folder with restricted permissions for only administrative or root use. One piece of advice: you should probably do only one or two things at a time and then reboot and be sure all is well until you are very comfortable with the OSX system.

5.11 Granting Privileges

In order to get around the problem of having no local accounts, I developed a simple *Perl* script that consults a remote MySQL table at login, checking the user login name (which is conveniently passed in from the *loginhook* call) and reading off a list of privileges that a particular user might need. In this way, I have dynamic control of how logged-in users can be granted, for example, *sudo* access. With this method, it is a relatively simple procedure to then login as that special user, make changes (including, if need be, the changing of the special password files to allow full administrative login) and then log out. I have not encountered any problems with this approach.

5.12 Refresh & Lost and Found at Login

As noted above in the section about duplicating the generic user home directory, a simple shell script does the real work of keeping this space clean. In effect, all of the files found in *writable* directories except the *~/Library/* directory are simply *dittoed* to the Lost and Found directory (you must use *ditto* to preserve the data correctly).

Once there, however, we do not want to start accumulating a ton of lost data! To get around this, we use the powerful UNIX *find* command in this same shell script to locate and remove all files older than a certain date (I use 7 days). This keeps things manageable, and allows users to easily locate missing work if they are prompt.

6. PREPARING THE MASTER IMG FILE

To ready the master machine as an image to build clones from, we use Carbon Copy Cloner. The simplest way to do this is to first prepare either a bootable CD for OSX (use the freeware utility *BootCD* found at various sites, such as www.versiontracker.com) or use your FireWire drive and setup a partition on that drive to be a bootable system. If you do the latter, note that you must be able to select the Startup disk, which requires administrative access. It makes sense to create this bootable partition early on in the process, before you install many software applications (since you are unlikely to need them for cloning!). In either case, be sure to include Carbon Copy Cloner and NetRestore on this bootable drive or CD.

Once you have a bootable medium that is *not* your master disk, boot the machine to it and run Carbon Copy Cloner. The settings are simple for this application. Set to not verify source, and do not repair permissions on source drive. Set to 'Create Disk Image on Target' and 'Prepare for Apple Software Restore'. The 'Make Bootable' will be checked on after these two are checked on. Note that when you click the 'Prepare for Apple Software Restore' checkbox, another dialog appears. I check the 'compress read-only image' option on to get an image file significantly reduced in size. Select the source (this is the local drive you worked on as the Master image) and the target – FireWire drive or Partition on local drive. I do not recommend a network share because it can be very slow. Note that you need twice the space to do this on the target as the final size. When you are ready, click the padlock, enter the administrative password, and then click the 'Clone' button. This process can take a long time!

7. CLONING

After Carbon Copy Cloner has created the master *img* file, move it to a portable or network accessible location if it is not there already. To make a clone, use the same bootable medium – either a CDROM or the FireWire drive – to boot to the clone-to-be¹. Next, start NetRestore. This is a very simple application to use. One small issue that needs attention: setting the preferences in NetRestore does not actually retain your changes from one boot to the next. You have to edit the XML control file to make this happen.

After NetRestore starts, locate your *source image* – hopefully on the booted medium or a partition. Drag this into the text entry field of NetRestore to set the name of the source it will use. Select your target – the local drive. Check ‘Erase Target Disk’, ‘Verify Restored Disk’ and ‘Set Target as Boot Disk’ to *on*.

Open the *Advance Options* under the File menu, and then enter the complete name (including pathname) of any post processing scripts you might want to use (to reset the entries in the *ByHosts* files, for example). Do not fiddle with the Buffers settings.

Finally, click the padlock, enter the administrative password, and then click the ‘Restore’ button. If all goes well, a typical locally hosted source image will take between 5 and 12 minutes to fully restore, a network image about twice that time. When done, NetRestore throws up a small dialog box with the time. Caution: if that time is ridiculously small – for example, .005468889999 *seconds* – you have not cloned the machine! This could be true if the master image is faulty or if you used the wrong source image.

If all went well, you should be able to simply reboot the machine and it will automatically boot to the clone drive and allow you to login as an authenticated user!

8. RESTORING DAMAGED MACHINES AND GOING FURTHER

If a machine really goes south, you might have to re-clone it using the same methods above. At some point in the future, I plan to deploy a refresh tool – possibly *radmind* – to make a rebuilding a little easier. This should make it possible to have unattended “repairs” that do not require hands-on attention.

9. CONCLUSION

This paper has tried to outline – albeit very briefly – the considerations you need to take in order to successfully deploy Mac OSX in an open access or lab setting. There are a myriad of areas that are useful additions to this process which I do not cover here at all. OSX presents a daunting challenge to the initiate. For experienced Macintosh users, the process may seem an exercise in frustration at first. With some patience and the use of the many

terrific resources out on the web, you too can “wrestle with alligators” and master Macintosh OSX with relative ease and far fewer headaches.

An expanded version of this paper may be obtained by contacting the author.

10. USEFUL RESOURCES

The following list of web resources is a small sample of the kinds of things you can find on the web for OSX. One problem to watch out for – there are many resources that reflect very early knowledge of OSX and have *not* been updated at all.

This is an essential stop! Be sure to read the Forums:

<http://www.macoslabs.org/>

One stop shopping for essential tools:

<http://www.bombich.com>

A good place to start a search for MacOSX stuff.:

<http://www.versiontracker.com>

Good introduction to LDAP and basic concepts:

<ftp://ftp.kalamazoolinux.org/pub/pdf/ldapv3.pdf>

This is the open source currently in use here at UVM:

<http://www.openldap.org/>

Fundamentals about the Open Firmware technology:

<http://www.firmworks.com/>

Fundamentals from Apple on Open Firmware:

<http://developer.apple.com/technotes/tn/tn1061.html>

Setting up Open Firmware protection using Apple’s utility:

<http://docs.info.apple.com/article.html?artnum=106482>

Download the GUI utility for OF from Apple:

<http://www.apple.com/downloads/macosx/apple/openfirmwarepassword.html>

Really good general overview article of security issues for UNIX:

<http://tr.sans.org/mac/freebsd.php>

Improving the Security of a Default Install of Mac OS X (v10.1) but still useful:

http://tr.sans.org/mac/default_install.php

Mac OS X 10.1.4: Security Analysis and Recommendations:

http://tr.sans.org/mac/osx_analysis.php

¹ Note that you can take several different approaches here since you obviously would prefer not to go through the entire setup process for a new machine. If you prefer, you can set the boot device in Open firmware with the *setenv boot-device* followed by the correct value. If you are cloning a lot of the same machine types, you can get this value from your master machine, then on the clone, enter Open firmware, set the boot device, the password and the security mode, and then boot to the required media that houses the image and NetRestore. If this is not a viable approach, you can at least reduce the number of steps for a brand new machine.