Learning Linux in a Windows Laboratory

Edward Crowley
Information and Logistics Technology Department
University of Houston

Abstract

Active learning experiences based upon Linux, and related open source tools, can enhance many courses. For example in a network course, use of sophisticated and free tools such as protocol analyzers, network monitors, and intrusion detection systems can enable students to actually capture and analyze packets as they transverse a network. Since many institutions have standardized on Windows for Computer Laboratories, implementing Linux based learning experiences may initially seem difficult. However, with a Live Linux CD, you can easily run Linux in a Windows Computer Lab.

Linux is a free operating system, generally packaged into distributions that can contain many useful tools. In addition to programming and scripting languages, Live CDs often include tools such as protocol analyzers, network monitors and network mappers, as well as conventional Linux utilities. In a Live CD format, these tools can be freely and easily distributed.

A Live Linux CD is a portable Linux distribution that boots and runs from CD, DVD, USB memory stick, or similar device. Our lab activities require no pre-lab setup and do not impact the Windows Systems upon which they run. That is, when the students power down the systems and remove their Live CDs, the Windows Systems are in the identical state that they were prior to the laboratory.

Consequently, our Linux laboratory modules require neither dedicated hardware nor dedicated support. For example, we have conducted Live CD based Linux Tutorials in Windows Labs at the New Jersey Institute of Technology, at the University of Houston, and at Lone Star Community College. We've also conducted Linux tutorials in hotel conference rooms with attendees booting their own Windows portable computers with Live Linux CDs.

In this paper, we will discuss how we have utilized Linux, Linux Distributions, and Live Linux CDs in our active learning activities. In addition, we will relate what we have learned from creating and conducting our active learning experiences.

We first began to utilize Live Linux CDs due to the limitations that we found in our existing standard Windows Laboratories. The following section elaborates on some of the issues that we encountered.

Introduction

Academic computer laboratories can be viewed from several different perspectives. For example, one can view laboratories from academic, administrative, or technical perspectives. Development of successful laboratory based learning activities requires an understanding of these differences. Consequently, we will first briefly present some of these different perspectives.

From a technical perspective, a homogenous computer laboratory standard can provide a focus for finite technical support resources. That is, a homogeneous laboratory standard may be the most economical way to support common learning activities.

From an administrative perspective, an organization may already have invested Windows or Apple training for their support staff(s). Existing support staffs likely have established the relationships with their outside vendors. And they may lack the skill and/or desire to support anything different than their existing standard software configurations.

When considering administrative costs, it is important to understand both initial (capital) and ongoing (support) dimensions. From a capital cost perspective, an organization may lack sufficient capital resources to support multiple heterogeneous laboratories.

Following this logic, many academic institutions standardize their laboratories on homogenous Windows (or Macintosh) systems. And while there are positive aspects associated with a single laboratory standard, from an academic perspective are also aspects that are other than positive. To optimize the design of an active learning experience, it is important to understand the subtle problematic issues associated with proprietary standards. In specific, three issues are:

1. Software compatibility and availability
2. Budget related administrative issues
3. Entrenchment of the status quo

One example of a compatibility issue is Microsoft's Word 2007 new default file format. Specifically, the new default file format (docx) cannot be read by previous office versions. Consequently, students utilizing older Office versions can have difficulty collaborating with students using the new version. In situations where students hand in work electronically, such as a distance education environment, this can be a particularly vexing issue.

One example of an availability issue is Microsoft's recent marketplace withdrawal of Windows XP. This withdrawal has led to classes and work groups where, on their personal laptops, some students have Windows XP and other students have Windows Vista. These compatibility problems can lead to collaboration issues. At the same time, budget issues can also be problematic.

Given the current economic climate, budget related administrative issues can be particularly challenging. In a specific academic environment, budget issues have both a time and a financial dimension.

In most academic environments, the acquisition of proprietary software is a multiple step process. For example, even if my Chair approves a software purchase, the purchase also requires approval from the College Dean. And if sufficient funds weren't allocated in the previous year's budget, the purchase may not be approved. This effectively extends the planning horizon for new proprietary software to over a year.

However, even if a purchase is budgeted, approved, and funded, the software still may not be available for a substantial amount of time. For example, an approved purchase request must travel from the department, to the college, to the university purchasing department and finally to an outside vendor. And prior to deployment, support requests must be submitted in a timely manner to provide sufficient lead time for installation, configuration, and testing.

Consequently time, in addition to dollars, can be a serious issue when dealing with proprietary software. While the particular time will vary among institutions, my experience is that there can easily be six months from the date of purchase approval till the date that the software can be utilized. But, from a student perspective, there is also another cost.

There is an inverse relationship between how popular a particular software is and the amount of open lab time available for students to work with the software. That is, as the number of sections of a particular class is offered increases, the amount of time the laboratory is available for student open lab use decreases.

At the same time, the more specialized and/or expensive the software is, the less likely that students are to have it already installed on their own systems. This creates a situation where students may become dependent on their educational institution for access to that particular software package. For example, for a particular class that utilizes an expensive proprietary software package as the number of class sections offered increases, the time available for open laboratory decreases. That is, there is an inverse correlation between laboratory need and availability. Consequently, time for student learning activities may be constrained by the lack of open lab time.

In a similar way, legitimate academic learning goals may require access to a computer laboratory. However, support resource constraints often produce laboratories with restricted Windows configurations i.e. limited rights, limited privileges, limited application software, and/or limited utilities. Utilized properly, these constraints help provide assurance that the laboratories will be available for future classes.

Though, existence of these constraints also means that certain activities cannot be implemented within those laboratories. Over time, these constraints can become a source of tension. Specifically, instructors may perceive restricted software configurations as restricting the scope of academic goals.

Fortunately, there are solutions. Later, we will discuss a solution that frees the instructor from the time constraints as well as from administrative budget cycle constraints. In addition, the proposed solution empowers instructors to optimize student laboratory learning experience. But first, let's look at several desirable solution attributes.

Solution Attributes

Given the current situation, there are several highly desirable solution attributes. Here are three of those attributes:

1. In both time and cost terms, the solution should free the instructor from budget related constraints.
2. The solution must free the instructor to choose an appropriate software version as well as to work free from the constraints of proprietary file formats.
3. The solution must work on previously configured Windows systems without impacting the existing configuration or the ability of future classes to utilize the existing systems.

Fortunately, there is a readily available solution that is congruent with these three attributes. That solution is to utilize Free and Open Software (FOSS) on a bootable Live Linux CD. The next section elaborates on both FOSS and Live LinuxCDs.

Free and Open Source Software (FOSS)

Free software has several popular definitions. For purpose of this paper, we will use the definition from the Free Software Foundation's (FSF) [2]. That is,

> Free software is software that gives you the user the freedom to share, study and modify it.

The FSF is the primary sponsor of the GNU project. In 1984, the GNU Project was formed to develop a complete Unix-like operating system. While the project was successful in developing utilities and related application software, the GNU project never completed a system kernel. Consequently, selected GNU utilities and selected applications are packaged with the Linux kernel to form different Linux Distributions. Different distributions are then released under a variety of software licenses including the GNU General Public License (GPL).

The GPL gives every person who receives a copy of a work permission to reproduce, adapt or distribute the work as long as any resulting copies or adaptations are also bound by the same license. That is, future works are restricted to being released under the same license conditions as the original GPL code. In contrast to copyright, this aspect of GNU philosophy is known as copyleft.

Not all open source licenses are considered restrictive like the GNU GPL. For example, the Berkeley Software Distribution (BSD), or BSD Style, license is considered a permissive open source license. Compared to other free software licenses, BSD licenses have few restrictions and do not support the GNU copyleft philosophy. Because the BSD License allows proprietary commercial use, there is widespread BSD code use in commercial products such as Mac OS X.

Note also that a particular software package may be free without being open source. The Adobe Flash Reader is an example of a program that is free but not open source. Wireless card vendors that offer proprietary binary card drivers without the source code are another example of free but not open source software.

Distinctions between licenses and software packages are important because thousands of separate software packages may be combined with other components including the Linux kernel to form a Linux distribution. A particular class of distributions distinguish themselves by offering a complete Linux system that may be booted from a CD or similar media.

These distributions may be distributed through the Internet as ISO files. Or they may be distributed on a bootable media such as a CD, DVD, USB flash memory stick, or similar portable device. First, we will examine Linux and Linux Distributions. Then, we will examine Live CDs. Finally, we will see an example of how Live CDs can be employed in the academic environment.

Linux and Linux Distributions

A typical Linux distribution consists of multiple components including:

- Linux kernel
- GNU tools and libraries
- Additional application and utility software
- Windows manager and desktop environment
- Applications and utilities
- Documentation

Linux distributions combine specific collections of utility and application software designed to appeal to specific audiences. For example, some distributions offer fully-featured desktops whereas other distributions may be server software focused and still others may be security focused.

Linux Distributions may be released under a variety of Open Source Licenses.
For example, the Knoppix Live CD uses GPL V2. Other distributions may contain free but not open source software. Within a particular distribution, different licenses may apply to separate software packages. Philosophical differences also impact specific distributions. Table 1-1 presents several popular Live CD distributions.

Table 1-1 Live CD Distributions

Live CD	Description
Knoppix	Based upon Debian, Knoppix is generally considered the oldest and best documented Live CD. Knoppix contains many system and security tools including the Wireshark Protocol Analyzer and Nessus vulnerability scanner. According to a 2006 Internet based survey, Knoppix was rated as one of the top 5 security distributions. [5]
BackTrack	Slax based, BackTrack focuses on security in general and penetration testing in particular. It simplifies complex configurations, including an automatic Kismet configuration and one click Snort IDS setup. According to a 2006 Internet based survey, BackTrack was rated as the top security distribution. [5]
Slax	Based upon Slackware, Slax provides a broad software collection, while maintaining a cd image small enough to be written to a 185 MB CD. For the default interface, it uses KDE. It is modular in design and can be expanded through the addition of modules. It is easy to customize and even has a Windows based wizard available for installing the distribution to a bootable USB stick.
Damn Small Linux (DSL)	DSL originally based on Model_K (a Knoppix variant), contains a complete desktop that only requires 50 megabytes of storage space. It uses FluxBox as its GUI. It is designed to be easily customized. It has minimal hardware requirements and supports older hardware well.

Live CDs

Because Live CDs dynamically configure the hardware and can save their settings to a USB memory stick, or other transportable media, the utilization of Live CDs enables the same physical computer to host multiple class sections without increasing the need for dedicated lab support. As a byproduct of the dynamic configuration process, the Live CD is transportable between different hardware systems. For example, a student can make changes to a project in a laboratory, save those changes, and revise those changes later on his dormitory computer.

Live CDs provide an expedient way for students to gain experience with Linux and with Linux based software tools. Our students have responded very positively to the utilization of open source tools and Live CDs. Anecdotally, many students report a very high perceived value added to the classes in which Live CDs are utilized.

Students are free to duplicate and distribute the Live CDs. Consequently, students may also utilize the Live CDs at home or at other locations. For example, our graduate students have used their Live CDs to support presentations to local and regional professional organizations.

Summary Freedom and Empowerment

Our experiences can be summarized in two general categories: Freedom and Empowerment. To discover that we could conduct our labs in any laboratory without interfering with the previously installed software was an enlightening experience. Student response has been enthusiastic. Tables 1-2 and 1-3 summarize these results.

Table 1- 2 Live CD Freedoms

Group	Freedom
Students	To continue their work at home, at work, or at other places.
	From needing to use the school's computer laboratories.
Instructors from	Their school's budget and purchasing cycles.
Students and instructors to	Utilize whatever software best meets their educational goals.
Schools from	The need to provide open lab time
	The need to track software licenses
Data from	Proprietary formats

Table 1-3 Live CD Empowerments

Group	Empowerment
Students to	Gain experience as computer administrators without the possibility of impacting other students or classes
	Legally distribute laboratory software with their friends or coworkers.
Instructors to	Choose software without regard to budget or purchasing constraints.
Schools to	Use laboratory hardware for multiple purposes rather than for a single dedicated purpose (or class).

Conclusions

When we began to employ Live CDs in our classes, we were concerned with how the students would react to them. We were especially concerned because our students had previously worked exclusively with standard Windows laboratories. We also were concerned because our antidotal experiences led us to believe that our students did not to have much exposure to command line utilities.

However, our experiences with Linux Live CD based learning activities have been very positive. For example, our exit surveys indicate that our students view the Live CDs, and related tools, very positively. They also value the fact that they can repeat their experiences at home or demonstrate their laboratory activities in a work environment.

Our plans are to create more Live CD based learning modules. In the future, we plan to also utilize some of the free Web services to create an online assessment portfolio.

Sample Netcat Laboratory Module

Netcat, a multifunction Linux utility, enables you to read and write data across an IP network using either TCP or UDP. Transported data may be in character format or it may be in file format. With Netcat, you can also perform a variety of other functions including network reconnaissance such as banner grabbing.

By design, Netcat functions much like the Linux "cat" command. The cat command is a filesystem utility used to concatenate or to print files.

Netcat facilitates multiple types of network functionality. For example, it can be used to transmit or receive data from any TCP or UDP port to any other TCP or UDP port. Netcat can even take data from standard input (keystrokes or data piped from a program) and sends the data across the network.

As you would expect, Netcat can be executed a number of ways. For example, you can execute Netcat manually from the command line. You can also call Netcat from a program or a script. Since it can create most any type of connection, Netcat can be employed to solve a wide variety of problems.

Goals
At the end of this module, you will be able to:

1. Describe Netcat's syntax.
2. Grab and analyze remote TCP/IP Service banners
3. Transfer data from standard input and output across a network
4. Transfer files across a network
5. List, define, and utilize two Netcat file transfer modes (push and pull).
6. Use Cryptcat to securely transfer files across a TCP/IP network.

Process
Boot your system with your Knoppix Live CD.
Open a Konsole window.
Then, get help with netcat (nc) from the man pages. Note that you can also get online help. [1]

```
knoppix@Knoppix: ~$nc –h
```

```
What command syntax would you use to make your computer a netcat
listener?
```

_____

```
What command syntax would you use to connect to a remote netcat
listner?
```

_____

Explain the function of the following three Netcat options. Give examples of where each option might be useful.

```
-e program
-u UDP mode
-v [or –vv]verbose
```

_____

_____

_____

Research online Netcat shell shoveling. Explain why it is dangerous.

_____

Banner Grabbing

Banner-grabbing is an information gathering exercise. The operative theory is that the more information that you can gather about a target, the more likely you are to locate a vulnerability. In this exercise, you will use Netcat to send a specific text string to a target. This string is designed to provoke a response. If the text string finds open target ports, the target will respond. By default, the response will be printed to standard output (your screen).

Here, we will use MIT's web server as a target. Though, you could use any web server. Now, in the Konsole Window type:

```
nc mit.edu 80
OPTIONS / HTTP/1.1   Enter
127.0.0.1
Press Enter Twice
```

Or

```
nc mit.edu 80
HEAD / HTTP/1.0
And press Enter Twice
```

Do you think that you can grab banners for other services? In your answer, list two other services and their ports.

_____

Data Transfer with Netcat: Two Modes Push and Pull

You can transfer data with Netcat in either of two modes: Push or Pull. In each mode, you will set up one computer as a Netcat Client and one computer as a NetCat Listener. This exercise will require good communications with your lab partner as well as precise syntax from you. Note also that the term "Client" is used here independently of the normal "Cient/Server" context. That is, depending on the mode a Netcat client may actually be functioning as a server.

For push mode transfers, one Netcat system is configured as a listener. Netcat listeners send data gathered from the network to standard output, which could be displayed on the screen or piped through another program.

When pushing a file you set up a Netcat listener on the destination system. This system listens on a specified port and dumps its output to a file. On the source system, Netcat is used in client mode to make a connection to the listening machine on the specified port, directing the file to be transferred as input. For example:

Table L1-1 Push Mode

Mode	Command String	Function
Listener	nc -l -p 1234 > FileReceived	Receives File
Client	nc RemoteIP 1234 < FilebeingSent	File Source

In contrast, for pull mode, you configure the sending machine in Netcat's listen mode. At the same time, you redirect the file being sent to Netcat's input. The destination machine is set up in client mode.

When the Netcat client on the destination machine connects, the file is dumped from the source system to the destination system. The destination machine can even pull the file by using a Web browser pointed to the appropriate port number.

Table L1-2 Pull Mode

Mode	Command String	Function
Listener	nc -l -p 1234 < filebeingsent	File Source
Client	nc remote_machineIP 1234 > outputfile	Receives File

For this exercise, we will work in two person teams. To complete the lab successfully, you will need to communicate well. You success depends on both your and your lab partners ability to communicate as well as ability to follow instructions.

We will start a copy of netcat on both machines. One machine will be yours. The other machine will be your lab partners.

```
knoppix@Knoppix:~$ netcat -l -p 5555
```

Here, using the -l switch, we are able to specify that netcat should go into listen mode i.e. to listen on the specified port. Using p 5555, we specify that we are using port 5555. On this

computer, netcat will sit and listen for TCP connections on port 5555. It will print any data that it receives to the screen.

On Computer 2, we start netcat as:

```
knoppix@Knoppix:~$ netcat 129.7.XXX.XXX 5555
```

This will connect to Computer 1, host 129.7.XXX.XXX on port 5555.

Now, on Computer 2 type:

```
This message was typed in Computer Two
Now I'm going to end communication with ^C (Ctrl-C)
knoppix@Knoppix:~$
```

After you press control –C, the connection is broken.

Since we are using a BASH shell, we may pipe (|) and redirect (>, >>, <, <<) data to and from a computer with netcat.

In the next steps, we will demonstrate how to transmit a file with netcat and redirection.

On Computer 1, start netcat as:

```
knoppix@Knoppix:~$ netcat -l -p 5555 > outputfile
```

This will run netcat with the same parameters specified previously, except now, it will redirect all text received into a file called “outputfile”.

On Computer 2, we will create a plain text file as follows:

```
knoppix@Knoppix:~$ ifconfig > infile
knoppix@Knoppix:~$
```

Here, we have written our Ethernet Configuration to a text file. This is the file we are going to transmit:

```
knoppix@Knoppix:~$ cat infile | netcat 129.7.XXX.XXX 5555 -q 10
knoppix@Knoppix:~$
```

Now to demonstrate that this has been transmitted to Computer 1, type:

```
knoppix@Knoppix:~$ cat outputfile
```

and you should see the configuration information from Computer 2.

And here we can confirm that it has. The -q 10 in the command line will quit after EOF (Otherwise netcat will hang waiting for more input for cat and we will have to terminate it manually). The parameter 10 causes it to quit after 10 seconds anyway.

References

[1] Crowley, E., Information System Security Curricula Development, Conference on Information Technology Education, SIGITE, West Lafayette, IN, 2003.
[2] Free Software Foundation, Retrieved Dec 09 from:
http://www.fsf.org/
[3] Granneman Scott, Live CD paradise, The Register, 7th May 2005, retrieved Dec 08 From:
http://www.theregister.co.uk/2005/05/07/live_cd_paradise/.
[4] Matthews, Jeanna, Hands-on Approach to Teaching Computer Networking Using Packet Traces, Conference on Information Technology Education, SIGITE, Newark, N.J., 2005.
[5] “Top 5 Security-Oriented Operating Systems”, Retrieved Jan 09 from:
http://sectools.org/sec-distros.html
[6] "The LiveCD List", Retrieved Dec 08 from:
http://www.livecdlist.com/
[7]Walden, Chris, The Distro Jungle, 28 Jun 07. Retrieved Dec 08 from:
http://www.ibm.com/developerworks/linux/library/l-linux-distros/

EDWARD T. CROWLEY
Professor Crowley serves as an Instructional Associate Professor in the College of Technology at the University of Houston. His research interests include the application of open source software in enterprise security areas, including wireless networking and intrusion detection. Professor Crowley is a member of the Information Systems Security Association (ISSA) and the American Society for Engineering Education (ASEE).