Intro to Linux Virtualization

April 11th, 2019

Today's Goals

- Virtualization
- Introducing Linux
    - Unix and Linux
- Navigating the Command Line
- Linux permissions and SetUID
- Aptitude (sudo apt install l1nux_kn0wl3d63)
- Git (git clone gud)

Announcements

- See https://sites.google.com/whitehatters.org/wcsc/announcements for CTFs we will be playing this week
- WCSC's 24/7 CTF board is back! Check it out at ctf.wcsc.usf.edu

Reading Material

- WCSC's new member page
    - Installing Ubuntu on Virtualbox
    - Codecademy's Linux Command Line
    - Bandit Wargame
- Linux vs. Unix
- Managing Files using the Linux Terminal
- Linux File Permissions
- Linux Special Permissions

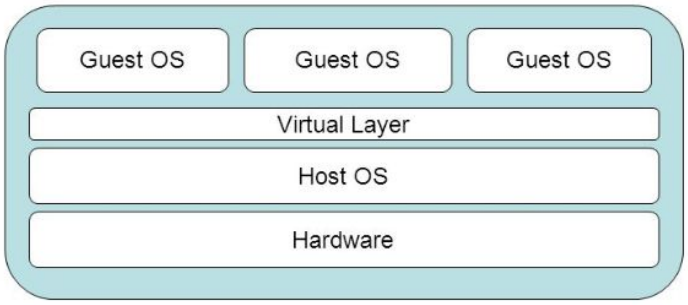
Why Linux Now?

For new members, the biggest hurdle is where to start. I should've started the semester with an introduction to Linux. Hopefully, this will encourage attending members to experiment with Linux themselves, and try out the demonstrations provided as part of the exploitation meetings. Plus, we now have a 24/7 CTF Board!

Note that this guide has lots of holes, since I don't really need reminders to talk about this stuff. Fortunately, their are great guides linked above and available via Google (Youtube is great).

Virtual Machines

Virtualization allows users to run an operating system (called the guest) inside of another operating system (called the host).



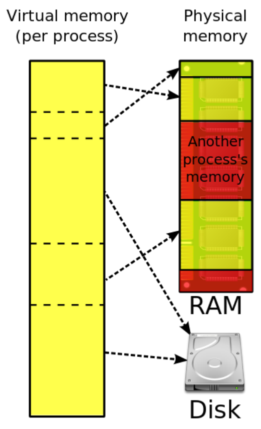
Operating System Virtualization

Virtualization

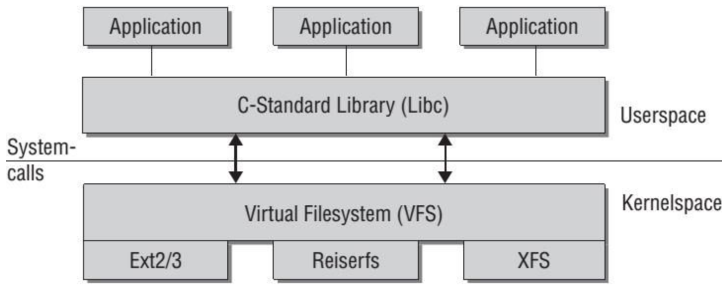
Virtualization is not new in computer engineering and science! Virtualization is built on the idea of abstraction, which is at the very core of every single concept in computer architecture in some form. When something is abstracted, the background or inner details are hidden, and we are concerned with only a few methods that are made available to interact with the abstracted object. Abstraction is easy to see in programming; when you call printf or gets, you don't care what it does beyond providing the correct arguments.

Virtualization occurs when we provide a virtual (i.e. software) implementation of the abstraction. We have virtual memory, virtual filesystems, application virtualization, data virtualization, network virtualization, virtual machines, etc. See https://en.wikipedia.org/wiki/Virtualization. In fact, virtual machines often make use of these other virtual devices, such as virtual networks, hard drives, and disk drives.

Why do I bring this up? Virtual machines are the logical following of abstracting operating systems. Virtual operating systems are not magic, but rather just computer scientists doing what they do best: abstracting.



Memory virtualization



Virtual file system

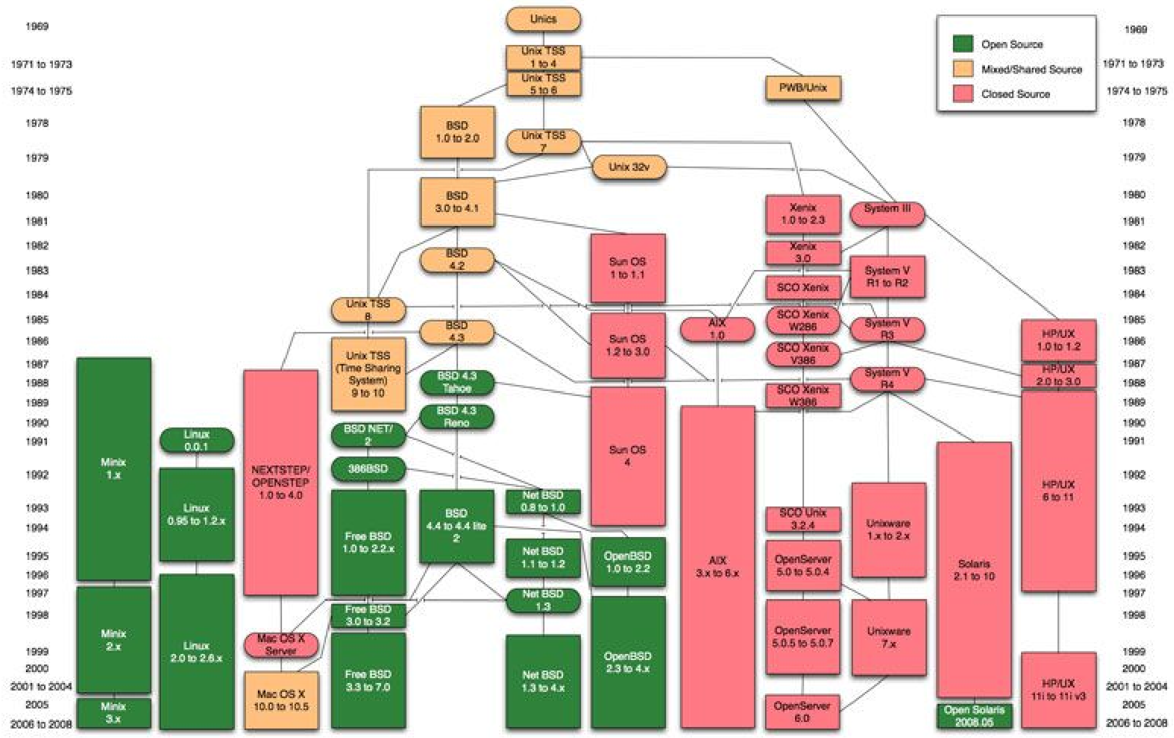
Virtualbox

I don't want to write anything on this, as plenty of guides exist, including the one linked in the Recommended Reading section. However, I will quickly walk through it on my machine for those in attendance.

If you are getting the error "**VT-x/AMD-V hardware acceleration is not available on your system**" or "**This host supports Intel VT-x, but Intel VT-x is disabled**" try following this guide: https://www.howtogeek.com/213795/how-to-enable-intel-vt-x-in-your-computers-bios-or-uefi-firmware/

Linux

Linux is a free and open source operating system that appeared in the very early 1990's from the efforts of Linus Torvald (developed the Linux kernel) and Richard Stallman (developed the GNU (GNU's Not Unix) set of utilities and programs). Together, Linux and GNU made a perfect pairing, and began to dominate. More about the history of Linux and Unix (a predecessor that was "mixed" source and inspired the development of Linux).



Since Linux is open source, new distributions (versions) of Linux sprung up, such as Ubuntu, Fedora, etc. Check out https://distrowatch.com/ for new Linux distributions and updates to existing distributions.

The Linux Command Line

As you can imagine, with hundreds of Linux distributions existing, there are differences between them, and these may change the commands you use to achieve certain tasks. However, the core set of commands should be very similar (very example, the set of GNU programs, like gcc), and once you've learnt one the others should be fairly easy to learn.

Let's open up the command line in the Ubuntu box we created and get started. The command line provides us access to all of the programs and features that Linux offers. Before we start testing things out, let's talk about the filesystem.

The Filesystem

The linux filesystem (ext4) is in charge of determining how files are interacted with (providing abstractions for opening, reading, writing, deleting, etc.). One of the primary functionalities of the terminal is to allow us to navigate and access the file system.

The two primary structures we will be interacting with are directories (folders) and files. In Linux, the file system is organized as a tree structure. Directories can contain files and other directories

(directories are thus recursive). The tree program (install using sudo apt-get install tree) provides us a diagram for this structure. Run the tree program by typing tree into the terminal.

To navigate the filesystem we will use the following commands:

- cd
    - Changes the directory that we are in
- ls
    - Lists the contents of the current (or provided) directory
- pwd
    - Prints the directory you are currently “in”

That’s really it! Let’s take a look at those, and some other Linux commands for managing files.

Linux Permissions

Now that we have navigated the filesystem, let’s talk about user permissions.

Run ls -l, and check out the extra output. Let’s break this down. (Again, sorry, I’m not writing this all down… Check out the reading material on Linux Permissions for a review or if you weren’t at the meeting).

SetUID

This is a pretty special linux permission, so we will talk about this. SetUID allows an executable file to be executed as the owner of the file. For example, if I create a file with my username kevindennis and run it as the user bob, the running program will have all the same rights and access as if kevindennis had run it.

This is important for a variety of programs. For example, the program passwd updates a user’s password. Every user on the system should be able to run this program, but this program will be required to access privileged files on the system. If we check out the passwd program, we see that it has setuid enabled, allowing the requirement above.

Why is this important to us? What if the passwd program, or another setuid program, had a vulnerability in it?

Sensitive Files in Linux

So, passwords (or, rather, their hashes) have to be stored somewhere, right? In linux, user information is stored in the /etc/passwd file, and the password hashes are stored in /etc/shadow. Let’s take a look at those files now!

Aptitude

Aptitude is the default package manager available on Debian-based distros (such as Ubuntu). As a package manager, aptitude provides several commands for easily installing, removing, and finding packages (software or programs). Let's walk through installing tree, the program we discussed before.

x86 on x86-64

A fairly common issue to run into when working on exploitation challenges is that the binary is 32-bit, but we have a 64-bit operating system. To get around this, you can use the following commands:

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install -y libc6:i386
```

Git

Not really related to Linux, but I wanted to discuss it regardless. Git is a distributed version control system for source code. This allows users to store and work on the same code base at the same time. Let's take a look at how to use git.

Why is Git important? Any software development team worth being a part of will be using Git in some capacity. Employers will expect you to have some familiarity with Git. Plus, many useful tools are available on Github, allowing us to use them to solve CTF challenges!