System Administration for Beginners

Week 8 Notes

April 13, 2009

1 Introduction

So far we have learned enough such that given a fresh install of an operating system, you would be able to turn into a complete Internet server. We have gone over the components that are required for most web services, where and how to obtain them, and you have a general idea of the factors to keep in mind wen choosing components for your Internet server.

As much as you've learned, however, setting all of this up is only half of a system administrator's job. The other half is managing the server and insuring that it remains secure. In this lecture and the one following, we will be taking a look at the extra precautions necessary when there is more than one user on a system and different methods to use when securing a system from local and remote attacks.

So far we have worked on a server that have either one user, yourself, or multiple trusted (hopefully) users, your project group members. We have not dealt with security too much other than working with permissions to get services running (`chmod`) and basic user security (`sudo`), mainly as a preventive measure to stop yourself from doing something you might regret.

When dealing with servers that have multiple untrusted users, there are two main ares that we want to focus on: resource sharing and security. This week, we will go over resource sharing and providing a general overview of security with regards to local users. Next week, there will be a more in-depth look at security, especially with regards to remote users.

2 Resource Sharing

A perhaps common conception of servers, seen especially in movies, are huge mainfraimes with unlimited resources, neat cabling, good cooling, etc. Unlike *The Matrix*, unfortunately, real servers have a finite amount of physical resources such as memory, processor power, and storage, as well as a finite amount of less tangible resources such as bandwidth. Each of these resources cost money, sometimes a one-time cost (like purchasing memory or more storage), otherwise a contiuous cost (purchasing bandwidth and power). Thus, it is one of the jobs

of a system administrator to make the most of these resources and ensure that they are shared fairly by all users.

The easiest solution to the problem of limited resources is not the purchase of the biggest and fastest server possible. If you have taken a computer science course, you may recall that an algorithm that takes n^n time does not double in speed when you double the processing power. It is not cost-effective to continuously purchase state of the art technology either. They will depreciate in value and performance very quickly (much like cars). Bigger and faster servers also have their own problems, physically. Most datacenters, for example, will charge services fees based on the size and power requirements of a server. Faster servers, besides a more expensive initial cost, will cost you more per month. They will also run hotter and are generally more prone to hardware failure.

In the end, the only universal solution to the problem of limited resources is careful monitoring and regulation by system administrators.

2.1 Resource Use and Abuse

Heavy resource usage can be justified or unavoidable. For example, if you are a system administrator for a research group, you might be administrating a server that performs complicated calculations continuously. The processor may always be overloaded, which cannot be helped. If this is the case, then you have limited solutions: optimize the calculation program, which may not always be possible, or upgrade the hardware.

Heavy resource usage soon becomes abuse when users are using more resources than needed, especially if the purpose of that server is not to perform heavy calculations or handle that sort of load constantly. There are multitudes of people who will abuse any freedom given. If you give users unlimited disk space, there is a very hig probability that someone will load your server with anything and everything they can find. It is then the system administrator's job to prevent such abuse.

The fuss about resource abuse is that running out of resources can result in a loss of data or business. If a server's hard drive space is exhausted, it will not be able to store any new transactions or information from users. If a server has used up its bandwidth, users will be unable to access the website hosted on that server. This latter problem is very important, as it has spawned a new category of attacks called Denial of Services (DoS) attacks, which online criminals use to extort online businesses.

Furthermore, resource abuse can affect other servers or cause legal problems. A user may abuse a server's resources to perform a DoS attack on another server. Not only does this waste the resources you purchased, but it could result in law enforcement agencies seizing your hardware. Similarly, abusive users may have taken over your server to host pirated data, which again drains resources and presents legal problems.

2.2 Quotas

An approach to preventing resource abuse is to impose quotas. Quotas are limits on measurable resources, such as disk space. UNIX provides multiple tools for managing quotas, which you will be working with during the laboratory this week.

Before you impose quotas on your users, you should develop a policy for resource usage. Having a pre-written policy provides a good insurance policy and helps you respond to disputes by users demanding additional resources without justification.

Draft your policy so that most users have exactly or just less than the amount of resources they need. In general, it is better to give less than what a user needs, as it is relatively easy to increase quotas, while decreasing quotas after the fact can result in technical and social problems.

Furthermore, resource usage tends to expand to fill a quota – if you give users too much disk space, they will probably find a way to use it. Giving users less resources than they need encourages them to use their resources wisely.

2.3 Accounting

Not all resources are managed with quotas. For example, processor usage and bandwidth are two dynamic commodities that are difficult to manage with quotas as they are either difficult to measure or have unpredictable usage patterns.

the best way to deal with these kinds of resources is to employ accounting, which is just a fancy word for monitoring. UNIX provides tools for auditing the amount of processor time and bandwidth that is used by a certain user or website. This information is then recorded in a log that can be reaad by system administrators. Since accounting deals with log file handling, we will take a look at it next week, under security.

3 Security

The second issue that affects multiuser systems is security. If you are the only user on a system, provided that you are the only one with access to it, you can be rather lenient with permissions. For example, you might grant 777 permissions on all your files and programs. Such a practice should be avoided, however, even on single-user systems as the compromise of a system could be devastating. Even if a hacker was able to get a limited access account on your system, they would be able to read and delete files with 777 permissions.

Assigning appropriate file permissions is extermely important on systems with untrusted users. For example, a client may store all their customer information, including credit card numbers, in a database file in their home directory. If permissions are incorrectly set on this database, a malicious user on the system could read the file and get access to this information.

While it is primarily the user's responsibility to ensure that they have the correct permissions on their files, it is also the system administrator's responsi-

bility to routinely perform file permission audits and adjust them as necessary to ensure the security of the system. `find` is particularly useful for these purposes as you can search for files by permissions. As an example, the system administrators that manage the instructional servers at Soda Hall regularly scan files for insecure files.

3.1 Access Control Lists

The file permissions that we have worked with so far have been rather crude. At the moment, you are only able to set specific permissions for the owner and group of a file, as well as provide a set of permissions for everyone else. With proper organization of users into groups, it is possible to build a very effective set of permissions. Unfortunately, such organization requires extensive planning, as UNIX permissions that we've worked with only allow one owner and group per a file or directory, so all the users that need access to a file or directory have to be in the same group if they are to share the same permissions. For systems where users are continually being added and removed, such planning is difficult or impossible.

Most UNIX operating systems now support an extended form of file permissions called Access Control Lists (ACL). ACLs allow users and system administrators to specify permissions for multiple owners and groups.

Do not become too dependent upon ACLs, though. They have been introduced this late in the course because it is important to become comfortable using regular UNIX permissions. You will find that in many cases, ACLs are really unnecessary; regular UNIX permissions are actually quite flexible. Furthermore, support for ACLs is not yet universal: the more popular UNIX versions and Microsoft Windows support ACLs, but some common tools do not support them, such as tar. Even so, there are more technical differences between varying versions of ACLs. The type you may be used to will not necessarily carry over to another system, while basic file permissions will.

3.2 Programs and their Privileges

In most operating systems, including UNIX-like operating systems, programs have their own set of privileges. Programs run with the privileges of the user who started them. If you execute a program, it will be able to read, write, or execute any other file or directory for which you have read, write, or execute permissions.

Last week, when you were configuring WordPress, you may recall that you had to modify the permissions on your WordPress file so that it would be read by anyone. You might haved noticed that this is rather insecure, since anybody would have been able to get your database password, but such a configuration is necessary because Apache, the web server, runs as its own user and group. This user, `www-data` on Debian systems, is not privileged, so it does not have access to any files or directories unless you grant explicitly grant it access with permissions. If you had not changed the permissions of the configuration file

to allow reading by anybody, Apache would not have been able to read the configuration file, and your installation of Wordpress would not have worked.

You can avoid this problem by using an ACL to specifically grant Apache access to the configuration file. However, this is only one solution to the problem of configuration for web services. Next week, a better solution will be covered, which comes at the cost of slightly reduced system performance.

For those of you who have been paying close attention, you may be wondering why Apache runs as the `www-data` user, even if you start Apache as root. Apache refuses to run as root because running as root is dangerous, just as is working as root. If a hacker manages to hack an Apache daemon running as root, they'll have root access to the system. Consequently, at the first chance possible, Apache will attempt to become a non-privileged user (through a mechanism similar to using `su`). When you start Apache as a normal user, Apache will continue to run as the same user, since it's already operating as an unprivileged user, but if you start Apache as root, it'll become `www-data` almost immediately.