CMPE 283 Worksheet 04

Project 04

Deadline 0900 Monday, October 20

Classwork

In this class we will continue setting up our droplets and Linux on our own computers.

Students who wish to set up a virtual server on a different provider from Digital Ocean or who want to use different operating system choices are absolutely free to do so. You should note that the instructions in the worksheet may not work exactly in other Linux distributions, though they will almost certainly work with slight modifications.

Any student who does not want to set up a droplet for money reasons, should contact me immediately so we can look for ways to solve the problem.

You should also note that your lecturer has not used Microsoft software at all for the past 14 years, so I will not have the knowledge to help you solve problems with Microsoft software, though you are, of course, free to use it if you have the necessary licences and are prepared to make the effort to support it yourself.

Setting up Linux as an alternative boot on your own computer will make it easier for you to experiment and learn offline and make certificated access to your droplet or other server easier. I strongly recommend you do this.

You should be able to do many of the tasks required from this course from a tablet or even a smart phone. However, the easiest and most productive way to do the practical work in this course will be from a personal computer with Linux installed. Solving the problems you will encounter with *any* method of accessing your server is *part of the course*.

Setting up Linux on your own computer

If you have not yet done this, do it now.

I have a couple of USB sticks with Ubuntu on that you can use in the labs.

1. Backup the data on your computer.
2. Boot Ubuntu from a USB stick. 64 bit verion is preferable on a modern computer, essential if RAM is more than 2 Gbyte.
3. Check that internet access and some common functions are working in the Ubuntu booted from USB stick.
4. If all is ok click "Install" and follow the on screen instructions.

Doing more to set up your droplet

I suggest you have Linux up and running on your computer while you are working on the server it will be handy, and also good practice.

If you do not yet have Linux on your computer, you will need to use a program like Putty to access your droplet.

Setting up a secure multiuser server

What we want to do In a typical apache setup, we will want to have more than one named web site on the server, and allow more than one user to update content on the server. We want all these to be as safe as possible.

So we want to be able to give our users the ability to freely upload material to one of the sites without intervention from the server administrator. However, we do not want those users to be able, deliberately or by accident, to access (read or write) in any way any of the other data on the server. This is true even if that user is the server administrator herself. Better safe than sorry, better to only have different hats with different capabilities and only use the super user hat when we really need to.

In general *n*x (Unix/Linux) systems allow read access quite generously. We want to block this for our web site creators.

We will do this by giving each website a user id. That user id will only be allowed sftp (secure ftp) access without command line usage and that sftp access will be in a "chroot jail". That means that the user will be restricted to accessing a particular directory and all its subdirectories and cannot even determine the existence of any other files on the system.

We also want to make our website set up as resilient as possible against online attacks by people we haven't given passwords to. To do this we need to close some of the more well known security holes that can occur in Apache set ups.

Finally, we know that we will have to come back and add new web sites from time to time. For each new web site we will need to go through the same process. We could write the instructions down on a piece of paper and follow them every time. An approach that requires a higher initial investment, but pays off as time goes by, is to write those instructions down in the form of a commented script. Then we can simply invoke the script to do the job, and, if the script is properly commented, and we need to do something different we have an example to start from.

So we have a fair number of new things to do all at once. Given that the principle is that everything we do to learn should also be immediately useful, this is unavoidable.

Before we start

Typing in the IP number to access our server is a pain, and now we have good reason for avoiding that pain. We want to set up several named hosts on our server. These will all have the same IP number, so our browser needs to send a host name along with the request. So we need to persuade our browser that the name we enter is actually associated with that IP number. Right now we are testing and do not want to buy any real domain names. We can create this association between name and IP number right on our own computer, without having to go through the whole DNS system.

To do this we just need to edit the /etc/hosts file on our own computer. Before looking up a domain name on the DNS on the internet, any request is first looked up in /etc/hosts.

Interestingly the exact same file, with exact same format, file name and even directory name is also present in Microsoft Windows systems, where everything is usually annoyingly different. The reason is that Microsoft did not want to believe in the internet. The first modern Windows, Windows 95, did not have internet support. The Microsoft Network was going to be Microsoft's monopoly answer to the internet. When this ruse did not work, Microsoft needed to add internet support very quickly. All of us were using third party add ons to get internet access. So they simply copied the entire internet stack from a version of Unix (OpenBSD) whose software licence allowed the lifting of code in this way. The GPL licence of GNU/Linux would not allow this. Terrified that the slightest change might stop it working, they copied it exactly the way it was, without touching anything. So /etc/hosts is a little island of Unix culture right there in the heart of MS Windows.

Find your /etc/hosts file. Easy on Linux. You need to search a bit to find it on Windows. I can't remember where they buried it.

Edit it. There are some lines that define names for 127.0.0.1, like this: 127.0.0.1

localhost After these lines add some lines for some fake domains that we will have on our server. ¡your ip number¿ mydomain.tk ¡your ip number¿ mydomain01.tk ¡your ip number¿ mydomain02.tk ¡your ip number¿ mydomain03.tk ¡your ip number¿ mydomain04.tk You can use any names you like. If you use real names, you will lose access to those real domains For future reference you should also go to your windows set up and set up these names there. At some point we will want to tests on our web sites using Internet Explorer or other browsers under Windows.

Now try logging in to the server.

You can log in like this

```
ssh -p1729 yourname@mydomain.tk
```

Now let us have a look at our default Apache setup

do

```
su
```

So, logged in as root:

```
cd /e<tab>/apa<tab>
ls -l
```

we can see one important config file - apache2.conf – along with some others, and a number of directories (the ones whose permission lists start with d). We need to read apache2.conf to find out how everything is organised.

To read a file while being sure that we will not change it by accident, we prefer not to open an editor. We can just feed the contents of the file into the less program which allow us to move backwards and forwards in the file without changing it.

```
cat apache2.conf | less
```

The lines in the file that begin with # are comments. The stuff at the beginning explains how all the files are gathered together to configure apache and points out that the details of how this is done are specific to Debian.

The comments that relate to how the sites-available and sites-enabled directories work are a little obscure.

The sites-available directory contains a separate config file for each site that we are planning to serve pages for. Each file has the the same name as the domain for the site .But we may find that a new site's config breaks the whole set up, or that we want to turn it off or on temporarily for some other reason. So the system is set up so that the main apache config reads in files from the sites-enabled directory. In the sites-enabled directory we simply set up links to the files in the sites-available directory. Deleting the link prevents apache reading the config for that domain, but leaves the file itself there for us to work on. Remembering the command to do the linking is hard, so we have a script do it for us. If we type (do not do this yet)

```
a2ensite www.mydomain.com
```

the a2ensite script creates a link in sites-enabled to the www.mydomain.com file in the sites-available directory.

If we type (do not do this yet)

```
a2dissite www.mydomain.com
```

The link is removed from the sites-enabled directory

We are gong to do most of this from a script, so you will need to read the script carefully to figure out what is going on.

Basic security

First let us use one of the files that make up the Apache config under debian to do some basic security The file is in the conf.d subdirectory. Remember that .d on the end of a name warns us that conf.d is a directory. In an environment where many names seem to have been assigned by an absent minded lunatic (/etc? Why?), we find a file with a logical name: security.

In the file are a number of security things we can do – mostly commented out so they are not active. We will just remove the `#` at the beginning of the appropriate lines to turn the security on.

ServerToken should be set to Prod, and except for the trace option, we turn everything else on. Now your security file should look something like this:

```
#
# Disable access to the entire file system except for the directories that
```

```
# are explicitly allowed later.
#
# This currently breaks the configurations that come with some web application
# Debian packages.
#
<Directory />
AllowOverride None
Order Deny,Allow
Deny from all
</Directory>


# Changing the following options will not really affect the security of the
# server, but might make attacks slightly more difficult in some cases.

#
# ServerTokens
# This directive configures what you return as the Server HTTP response
# Header. The default is 'Full' which sends information about the OS-Type
# and compiled in modules.
# Set to one of:  Full | OS | Minimal | Minor | Major | Prod
# where Full conveys the most information, and Prod the least.
#
ServerTokens Prod

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (internal error documents, FTP directory
# listings, mod_status and mod_info output etc., but not CGI generated
# documents or custom error documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
#
ServerSignature Off


#
# Allow TRACE method
#
# Set to "extended" to also reflect the request body (only for testing and
# diagnostic purposes).
#
# Set to one of:  On | Off | extended
#
TraceEnable Off
```

```
#
# Forbid access to version control directories
#
# If you use version control systems in your document root, you should
# probably deny access to their directories. For example, for subversion:
#
<DirectoryMatch "/\.svn">
Deny from all
Satisfy all
</DirectoryMatch>

#
# Setting this header will prevent MSIE from interpreting files as something
# else than declared by the content type in the HTTP headers.
# Requires mod_headers to be enabled.
#
Header set X-Content-Type-Options: "nosniff"

#
# Some browsers have a built-in XSS filter that will detect some cross site
# scripting attacks. By default, these browsers modify the suspicious part of
# the page and display the result. This behavior can create various problems
# including new security issues. This header will tell the XSS filter to
# completely block access to the page instead.
# Requires mod_headers to be enabled.
#
Header set X-XSS-Protection: "1; mode=block"

#
# Setting this header will prevent other sites from embedding pages from this
# site as frames. This defends against clickjacking attacks.
# Requires mod_headers to be enabled.
#
Header set X-Frame-Options: "sameorigin"
```

One of the changes we made here required that the headers module be enabled, so we need to enable it

```
a2enmod headers
```

then restart apache

```
service apache restart
```

Fingers ctossed for no errors as apache restarts

Now we should check from our browser that we can still access the server and get the default page.

Setting up the sftp jail

First we must set up an sftp group of users that will all live in sftp jails.

```
adduser sftp
```

Now we need to modify `sshd_config` to set up the jail. This involves adding a whole bunch of lines to the end of `sshd_config`, an error prone process. Here are the lines we will add:

```
#
# This is the set up for the sftp chroot jail  - add to the end of sshd_config
# Users who are members of the sftp group will be jailed to their home directories
# And their sftp activities will be logged, if and only if, all the necessary set u
# that is done by apachevirtualhostconfig has been done by that script or by hand
#
Subsystem sftp internal-sftp -u 0027 -f LOCAL7 -l VERBOSE

Match Group sftp
    ChrootDirectory %h
    ForceCommand internal-sftp -f LOCAL7 -l VERBOSE
    X11Forwarding no
    AllowTcpForwarding no
```

Open up `sshd_config` in an editor and just copy paste these lines from this document in at the end of the file.

You will then need to find the existing Subsystem sftp line in the file and disable it. You could delete that line, but putting a `#` character at the beginning of the line has the sane effect, means we can see what we did and makes it easier to go back if something goes wrong. If you want to be really fussy add another comment line above that line to say

```
# disabled to allow creation of sftp jail
```

You will need to use the mouse and right clicks to to the pasting.

Save the file and reload the ssh configuration

```
service ssh reload
```

We now have sftp jailing set up.

Test your sftp jail by using an sftp client to transfer files to the home directory of a user on your server. You will need to make a special user and make that user a member of the sftp group, and set permissions on /home and /home/youruser correctly. Ordinary ssh access will *not* work correctly for this user.

Because ssh will not work for this user you should prevent this user from using a shell. You can do this by setting the shell for the user to /bin/false. Find out how to do this.

Find out how permissions need to be set up on a home directory to allow sftp jailing over sftp.

Try and find out what you need to do and how to do it. Use Google, ask smart questions. This part is *deliberately* not written here. Finding out is part of your assignment.

Assignment

1. Finish up what you started in class. Be prepared to show what you have done in the next lab class.
2. In particular, next week you *must* be able to show your assistant that sftp jailing is now working on your server.
3. Write up an account of what you did and what you learned. What was wrong or missing in these instructions? Could you now explain what you have done so far to a friend?
4. post your account to online.bilgi.edu.tr along with a link to your droplet.