# Linux Servers Lab Guide

**Note: Know the answer to all of the questions before moving on! Google them first and if you still don’t understand, ask for help!**

You need 2 VMs for this lab (one will be your client, the other is the server). Make sure you set both VMs to bridged mode. Do as much of the lab as possible, but if it is too tiring for you, at least get accustomed to it by setting up the servers. The rest of the lab is up to you, but we suggest you test it out now or in your free time to understand the importance of secure configurations.

Pro tip: when troubleshooting, it’s always good to look at forums like StackOverflow! Other people have probably already found a solution to your problems

## Part 1: SSH

### Understanding SSH

1. Check if you have the SSH server installed on your server image. If you don’t install it.
   1. apt-get install openssh-server
2. Generate your ssh keys
   1. Where are your public and private keys stored?
3. Retrieve your IP address from the server image
   1. Make sure your VMs are on bridged mode
4. Establish a connection with the server
   1. Hint: ssh [username]@[ipaddress]
5. Logout of your ssh session
   1. Ctrl-d

### Using SCP

1. Create a new folder called files in your home directory in the client
   1. create testfile1 and testfile2 inside it
2. Try to use remote file transfer
   1. Hint: the related command is scp
   2. What are the three ways in which the scp command can be used?
3. Give that remote server your testfile1 and testfile2

### Set Up Key Based Authentication

1. Enable public key authentication on the remote server
   1. Hint: PubKeyAuthentication
2. Type the command to generate a key pair
   1. Hint: it has ‘keygen’ in the command
   2. Why would passphrases be recommended?
   3. Where are your keys stored (considering you kept the defaults)?
   4. Which file stores your public key? Which file stores your private key?
3. Make sure you set the right permissions on these key files.
4. Log into the remote server using the client
   1. Hint: look above for the syntax
   2. What type of authentication are you using now to log in?
      1. It’s not key based authentication … so its \_\_\_\_\_\_\_
5. Use the scp command to copy the public key to the remote server
   1. Hint: scp [file you want to copy] [username]@[ip-address]:[location you wish to upload it to]
6. Move the newly uploaded public key to the authorized\_keys file
7. Check your /etc/ssh/sshd\_config file
   1. Is public key authentication authorized?
   2. Is RSAAuthentication authorized?
   3. Is the authorized\_keys file authorized?
8. Restart ssh server
9. Now try to login to the remote server with SSH using public key authentication
   1. ssh -i [path to private key] [username]@[ip-address]
10. Disable PasswordAuthentication

### SSH Configurations

1. Navigate to your ssh configuration file in your server
   1. cd /etc/ssh/sshd\_config
2. Disable Root Login
   1. Hint: PermitRootLogin
3. Now test this setting yourself
4. Log in as root using ssh
   1. What happens?
5. Switch this setting back to ‘yes’
6. Now log in as root using ssh again
   1. What happens now?
7. Try out all the configurations we mentioned in the slideshow! Have fun! :)
   1. If you think this is busy work, just do a few more configs. We don’t want you to get bored and irritated about learning cyber.

## Part 2: FTP

### Installation

1. Do the following on your **server VM:**
2. Install the latest version of vsftpd
   1. How do you make sure that you have the newest one?
   2. Why is it important to have an updated version of the server?
3. Allow ftp through the firewall

### Setup

1. Do the following on your **server VM:**
2. Create a new local user named ftpuser. Set the password to “password”.
3. In /home/ftpuser/, create a new directory named project\_folder
   1. Make sure the owner and group are both ftpuser
4. In project\_folder, create a new file called download\_file. In it, write “wow you have successfully downloaded a file!”

### Configuring Access

1. Do the following on your **server VM:**
2. Edit /etc/vsftpd.conf
   1. Pro tip: much of this file has documentation in comments(#). Try reading this before jumping to google!
3. Do not allow anonymous ftp. However, DO allow local users to log in
   1. What would happen if you didn’t allow anonymous access and you ALSO didn’t allow local users to log in?
4. For this example, we’re going to allow ftp write commands.
   1. Note: different scenarios call for different configurations. Sometimes, you will need to turn this option off.
   2. What are ftp write commands?
5. Make sure that vsftpd will chroot the local user
6. Specify the chroot directory as /home/ftpuser/project\_folder
   1. Your default vsftpd conf probably doesn’t have this option. Have fun figuring this one out:D
7. Allow for writeable chroot
   1. Why?
8. Restart the service!
   1. Why is this always necessary after config? What happens if you don’t?

### Putting it in action!

1. Do the following on your **client VM:**
   1. Why is it important that we test everything with a client?
2. In your home directory, create a new file called upload\_file
3. In upload\_file, write “wow you have successfully uploaded a file!”
4. Set **both VMs** to bridged mode
5. On your **server VM**, find out what your IP address is
   1. What command do you need?
6. On your **client VM**, establish an ftp connection to your **server VM**
   1. Login as ftpuser
7. Set the local ftp file download directory to your home directory
   1. Pro tip: look up the lcd command!
8. List all of the files available
9. Download download\_file. It will go to your home directory
10. Upload upload\_file to the server
    1. Pro tip: always verify your results! Check that the file now exists on the server and still says “wow you have successfully uploaded a file!”
    2. Hint: When you ftp into a server, you don’t get a text editor. You also don’t get commands such as cat or less. What you CAN do is: get upload\_file -
    3. In this example, we allowed for writeable chroot. What would have happened if we didn’t allow for writeable chroot?
11. Exit out of your ftp connection
12. Verify that you successfully downloaded download\_file

## Part 3: LAMP Stack

### Installation

1. Install Apache2: apache2, apache2-utils
2. install Mysql: mysql-server, mysql-client
   1. Don’t forget your mysql password lol
3. install PHP and other necessary modules: php7.0, php7.0-mysql, libapache2-mod-php7.0, php7.0-cli, php7.0-cgi, and php7.0-gd
4. Verify that php rendering is functional
   1. Create an info.php file in /var/www/html/ which contains:

<?php

phpinfo();

?>

* 1. go to your browser and access info.php. You should see a page full of statistics
  2. Should you keep this file as a hostable file? Why or why not?

1. Download the contents of the latest wordpress version
   1. You can find it at wordpress.org/latest.tar.gz
2. Extract the file. You should be left with a wordpress folder. Move the contents out of the wordpress folder and into /var/www/html/
   1. Your /var/www/html should now have some files/folders such as wp-includes, wp-content, index.php
3. Remove the default index.html in /var/www/html/
   1. It will conflict with index.php
4. Change the owner and group of every file in /var/www/html to www-data
5. Change the file permission of every file in /var/www/html to 755
6. In mysql, create a new wordpress database called “wordpress”
7. Create a new wordpress user called “wordpressuser” and grant all privileges to the wordpress database
   1. Remember to flush privileges then exit
8. Rename the default wp-config-sample.php to wp-config.php
9. Edit wp-config.php
   1. Change the DB\_NAME, DB\_USER, and DB\_PASSWORD to match our new database and user
10. Restart your LAMP services
11. Allow all of the LAMP services through the firewall

### GUI time

1. Head to your browser and search for localhost
2. Select English and then continue to the next page
3. Set whatever site title, user, and password you want. Just put your school email for the section “Your Email”
   1. Don’t forget your user and password!
4. Install wordpress
5. Log in, and you will be greeted with the wordpress dashboard :D

### Playing with the wordpress dashboard

1. Create a new post. It can be whatever you want :D
2. Familiarize yourself with the updates page. You never want an outdated server!
3. Create a new page :D
4. Change the theme so that it is not default
5. Check out your plugins. Why is it a good idea to remove Hello Dolly?
   1. Pro tip: in security, everything that isn’t critical/necessary should BE GONE
6. Check out the user section. Make sure you know how to delete rogue admins.
   1. Add a new user and then delete them
7. Go to the general settings section. What’s wrong with the wordpress address and site address? What should they be instead?
   1. Remember: Make sure that your client and server VM are both on bridged mode!
   2. Instead of having http://localhost, you should have http://[server IP]
   3. You can find your server’s IP using the ifconfig command

### Checking your website

1. On your client VM, try accessing your new web server
2. Feel free to click around on different posts, etc.
3. Make some comments :D
4. Try logging into the dashboard from your client
   1. Hint: wp-admin

### Securing Apache

1. On your client, look for a page that doesn’t exist on your web server
   1. Ex: try looking for yeehaw.txt
   2. What does Apache output?
   3. Is it secure to reveal information such as port and service version? Why?
2. Configure apache2.conf so that no version information is leaked
   1. Don’t forget to reload your service!
3. Verify that step 2 has been implemented correctly
   1. If the service version still shows up, that means another configuration file is conflicting with it
   2. Hint: check security.conf
4. Try turning on TraceEnable
   1. Hint: check security.conf
   2. Why are we doing this configuration in security.conf instead of apache2.conf?
   3. Don’t forget… restart the service!
5. Try creating a trace request
   1. Does it work?
   2. Why is this bad?
6. Now turn it back off and try creating a trace request
   1. It doesn’t work now right? How can we tell?
7. In apache2.conf, configure your root directory to not allow for any options by default
   1. Why is having additional options bad?
   2. Will /var/www/html/ still be able to perform its additional options? Why?
   3. What do the <Directory> </Directory> tags even mean???
8. On your client, verify full site functionality

### Securing MySQL

1. Find the mysql configuration file
2. Check the bind-address. What does bind-address even mean/what does it do? What should it NEVER be?
3. Check your mysql user. What should it be?
4. Try adding the line skip-grant-tables under [mysqld]
   1. Now try logging into mysql with mysql -u root
   2. Don’t specify -p. What does -p do?
   3. What happens(or rather, what DOESN’T happen)? Is this secure?
   4. Yeahhhhhh take out skip-grant-tables pls……
5. For security reasons, it’s always good to have local-infile=0 under the [mysqld] section
6. On your client, verify full site functionality

### Securing PHP

1. What is expose\_php? Why and how should you turn it off?
2. Let’s make a webshell!!
   1. In /var/www/html/, create a new file called webshell.php
   2. In it, put the line: <?php system($\_GET['cmd']);?>
   3. On your client, access the webshell. Use it to run a command (like this):
      1. [ip addr]/webshell.php?cmd=cat /etc/passwd
      2. The stuff after the question mark just allows you to enter the arguments for the webshell to function
      3. What do you see?
   4. This webshell relies on the php function called system(). Disable it in php.ini as well as exec, shell\_exec, passthru, popen, curl\_exec, curl\_multi\_exec, parse\_ini\_file, show\_source, proc\_open, and pcntl\_exec
      1. All are commonly found in exploits

### Securing Wordpress

1. Things that you ALWAYS want to check through the dashboard:
   1. Updates
   2. Hosted content
   3. Plugins
   4. Users
   5. Settings
2. Click on these different dashboard pages to make sure you know what everything looks like/how it works(TBH it’s a gui...fairly straightforward)