**Objective**: Install PuTTY for Windows. Create public/private keys and configure PuTTY to launch SSH sessions using key-based authentication. Develop basic competence operating and configuring Linux servers.

**Discussion:** You will complete labs in this and other TAMU-SA computer information systems courses in JagCloud. JagCloud is a cloud-hosted environment with virtual Linux server instances. An instance is a single, unique virtual machine within a cloud environment. JagCloud instances do not have graphical user interfaces (GUIs). Therefore, you will access and interact with the servers using Secure Shell (ssh). Although ssh is native to most Linux distributions, ssh is not native in Windows. Windows requires a third-party ssh client. PuTTY is recommended for JagCloud access from Windows systems.

For security purposes, JagCloud instances may also have limited Internet exposure. As a result, you will need to access a private network within JagCloud to complete some labs. The private network will be accessed through a remote access VPN using Softether VPN client.

This lab will guide you through:

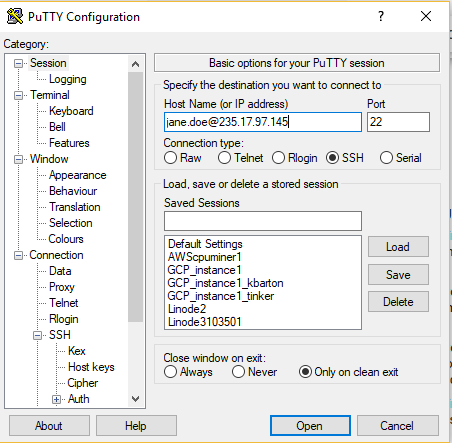
1. Installation of PuTTY and related utilities.
2. Creation of public/private key pairs for ssh authentication.
3. Sharing the public key with JagCloud administrators. JagCloud administrators will grant you access to instances in JagCloud.
4. Installation and configuration of Softether VPN client.
5. Accessing a JagCloud instance using PuTTY.
6. An introduction to Linux.
7. Transferring files using secure copy (scp)/PuTTY secure copy (pscp)

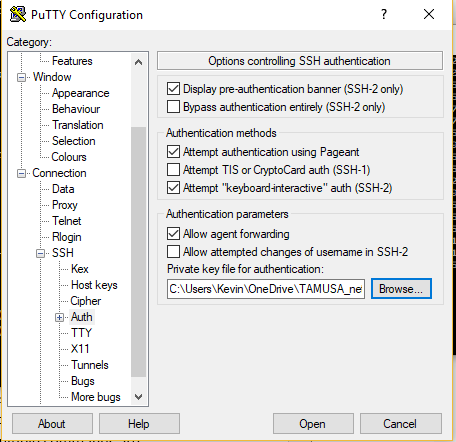
You will make two lab submissions. First, you will submit a public key. The public key will be used to grant access to JagCloud instances, and therefore the first submission must be done to gain access to JagCloud. **The public key should be submitted promptly**. **You cannot do subsequent steps until you have submitted your public key**. You will maintain lab notes throughout the lab, and build a file with lab results as you work through the lab. The second submission will include your complete lab notes and the results file you build during the lab.

**Resources**:

1. PuTTY and PuTTYgen
2. Softether VPN client
3. Storage device for public/private key pair

**Overview**:

1. Install PuTTY.
2. Generate and secure a public/private key pair.
3. Share the public key.
4. Install Softether VPN client.
5. Configure Softether VPN client.
6. Introduction to Linux.
7. **Procedures: Configure Softether VPN Client**
   1. Open Softether VPN Client Manager.
      1. Select **Add VPN Connection**. Edit the setting as follows:
      2. **Setting Name**: jagcloud
      3. **Host Name**: 35.202.29.108
      4. **Virtual Hub Name**: VPN
      5. **Auth Type**: RADIUS or NT Domain Authentication
      6. **User Name**: *firstname.lastname*
      7. **Password:** *Password is the first four characters of first name + first six characters of your Jaguar email username. Ex: If name is Jane Doe and email is jdoe099, the password is janejdoe09*
      8. Do not change any other settings.
      9. Click **OK**.
8. **Access JagCloud Instance using PuTTY**
   1. Open PuTTY
   2. **Host Name (or IP address)**: Enter *username@IP address* for the instance. Your username will normally be *firstname-lastname*. The instance is not Internet facing, you must first connect to JagCloud through the VPN.
   3. 
   4. Expand **Connection** and **SSH**. Select **Auth**. See illustration below.
      1. Select **Allow agent forwarding**.
      2. Browse to and select your private key file (i.e., *firstname-lastname*.ppk).
   5. Click **Open**. You should receive a terminal screen.



1. **Introduction to Linux**
   1. **Root**

The root user has privileges to create, read, write, edit or delete any file or directory in a Linux system. Root also has privileges to execute all commands on a Linux system. Administrators of Linux systems must be able to elevate user privileges to root, as well as understand the security risks associated with root privileges. This lab will demonstrate two methods to access root privileges, using the commands **sudo** and **sudo su**. The command **sudo** is useful when just one or two commands need to be executed with root, while **sudo su** is useful when multiple commands are expected. The prompt “$” indicates user mode (not root), while the prompt “#” indicates root.

The /etc/shadow file stores encrypted passwords and additional password information for all user accounts. Root is required to read /etc/shadow. We will use this file to demonstrate root function. You will read content from the shadow file and store it to a new file. Additional information will be added to file throughout lab, and the file will be submitted in Blackboard. You will also maintain and submit lab notes.

Complete the following commands and record the results in lab notes. The command **cat** is used to read the text from a file. The prompt **user@host:~$** reflects the user account executing the command, the hostname, and that the user is in user mode. We will also use the **ls** and **grep** commands and three types of pipes (**|** , **>** and **>>**). The pipe **|** feeds the output from one command into another. The grep command searches for a specified text string. The pipe **>** reads the output from the command to a file, while the pipe **>>** appends the output to a file without overwriting existing content in the file. **The file lab1-results.txt needs to be created in your home directory, so replace *your-username* in the file path below to your username. Do not copy “your-username”**.

user@host:~$ ls

user@host:~$ cat /etc/shadow

user@host:~$ sudo cat /etc/shadow

user@host:~$ sudo cat /etc/shadow | grep rmaddow

user@host:~$ sudo su

root@host:~# cat /etc/shadow

root@host:~# cat /etc/shadow | grep rmaddow > /home/your-username/lab1-results.txt

root@host:~# chown *your-username*:*your-username* /home/*your-username*/lab1-results.txt

root@host:~# exit

user@host:~$ ls

user@host:~$ cat lab1-results.txt

* 1. **Updates and Patches**

Most packages in a Linux distribution can be updated with the **update** and **upgrade** commands. The command **update** updates the database of patches and upgrades that are available, while **upgrade** installs those updates. This section will demonstrate how to list the distribution version and the available updates, install updates using **apt** and demonstrate that the instance’s updates are current. You will also append data to the file *lab1-results.txt* using the **>>** pipe. Execute the following commands. Report results in your lab notes. This section begins with the **echo** command. **echo** prints designated text to the screen. We will append that text to the file to designate results for each section of the lab.

user@host:~$ echo >> lab1-results.txt

user@host:~$ echo Updates >> lab1-results.txt

user@host:~$ lsb\_release –a

user@host:~$ uname –a

user@host:~$ lsb\_release –a >> lab1-results.txt

user@host:~$ uname –a >> lab1-results.txt

user@host:~$ apt-get upgrade --dry-run

The remaining steps require root. Elevate to root privilege.

root@host:~# apt-get update

root@host:~# apt-get upgrade –y

root@host:~# apt-get upgrade --dry-run >> /home/your-username/lab1-results.txt

root@host:~# exit

* 1. **Navigating Directories**

This section introduces the **cd** command and provides practice using **cd** with various shortcuts. The **~** represents the users home directory and can be used as a shortcut to move to the user’s home directory from anywhere in the directory tree. The shortcut **..** is used to move up one directory. Both shortcuts can be combined with other strings to move through the directory tree. You will also use additional options of the **ls** command and practice with the **pwd** command. The **pwd** command list the current directory. The tab button can be used to auto-complete directory and file names. Using the tab button not only speeds up navigation through the file structure, but also reduces the likeliness of typing or spelling errors. One thing … capitalization matters in Linux. Uppercase and lowercase always mean different things, do not interchange them. Things just will not work.

user@host:~$ echo >> lab1-results.txt

user@host:~$ echo Navigating >> lab1-results.txt

user@host:~$ pwd

user@host:~$ cd ..

user@host:~$ pwd

user@host:~$ cd /home/ubuntu

user@host:~$ sudo su

root@host:~# pwd >> /home/your-username/lab1-results.txt

root@host:~# exit

user@host:~$ cd ~

user@host:~$ pwd >> lab1-results.txt

* 1. **Finding Files and Directories**

You may sometimes struggle to find a file or directory. If it’s a system file or directory, a Google search can usually help. However, a Google search won’t likely help you find a directory created by a user or a data file. Fortunately, you can search the file system with **find.** This section will demonstrate how to find a file or directory using **find**. It will also introduce you to the **help** option and **man** pages. Most commands have a either **help** or **man** pages, or both. When all else fails, use a command’s help option or man page for guidance on how to use that command. Help will usually be shown using the options **-h** and/or **--help**. **Man** pages can normally be accessed using **man** ***command***. For example, **man** **pwd**. Exit a man page using the “**q**” key. The command **find** uses the –help option, not **-h.** The nmap command will be used to demonstrate the **-h** option. We will use the **find** command to search for the file lab1-results.txt. Report the results in your lab notes and offer an explanation why the different commands gave different results.

user@host:~$ cd /home/ubuntu

user@host:~$ man find

user@host:~$ q

user@host:~$ find –-help

user@host:~$ nmap –h

user@host:~$ find –name lab1-results.txt

user@host:~$ sudo find –name lab1-results.txt

user@host:~$ sudo find /home –name lab1-results.txt

user@host:~$ sudo find / -name lab1-results.txt

user@host:~$ sudo find /etc –name lab1-results.txt

The command **locate** is also useful for finding files. Locate maintains a database of all files. That database updates periodically, but can also be updated manually using the **updatedb** command. These commands are demonstrated below. Next, you attempt to find a file that we know exists. It may not be found. We then update the database, and use **locate** again to search for the file. The command **locate** is convenient because it searches the entire file system, not just the directory and subdirectories of current working directory.

user@host:~$ locate lab1-results.txt

user@host:~$ sudo updatedb

user@host:~$ locate lab1-results.txt

* 1. R**eading, Creating, Editing, and Deleting Directories and Files**

This section will introduce the commands **cat**, **nano** and **rm**. Command **cat** reads a file. Linux distributions ship with at least one text editor. Common text editors include **nano**, **vi**, **vim** and **gedit**. **gedit** is a GUI based editor, and is common in Linux desktop distributions. However, the JagCloud instance you are working with is a server, so does not support GUI based tools or applications. You will practice with **nano**. Note, you begin this section in a directory other than your home directory. Therefore, you must provide the full file path to write to files in your home directory. Include **~/** as a shortcut to the full file path of your home directory (the full file path would be **/home/*username*/lab1-results.txt, the shortcut is ~/lab1-results.txt**).

You will use a text editor, **nano**, to create and edit a new file. After reading the file with **cat**, you will delete the file using **rm**. Note, Linux does not recognize spaces in a filename without the use of quotation marks. For example,

**nano new file.txt**

would create two files (**new** and **file.txt**) in the current directory. The command

**nano ~/new file.txt**

would also create two files. The first would be **new**, located in the user’s home directory. The second would be **file.txt** in the current directory. The command

**nano ~/”new file.txt”**

would open a single file in the user’s home directory. Common practice in Linux is to exclude spaces in file or directory names.

Directories are created with **mkdir**. Files and directories can be hidden by placing a period before the file or directory name. You will create visible and hidden directories in this section.

user@host:~$ echo >> ~/lab1-results.txt

user@host:~$ echo Create/Edit Files/Dir >> ~/lab1-results.txt

user@host:~$ mkdir ~/newdir

user@host:~$ mkdir ~/.hiddendir

user@host:~$ ls ~

user@host:~$ ls –a ~

Compare the results from the two previous commands. Note the hidden directories and files are listed using the **–a** option with the command **ls**.

The next step creates and opens a file in the user’s home directory with nano

user@host:~$ nano ~/new-file.txt

Enter the following sentence:

**The quick brown fox jumped over the lazy dog.**

To exit nano, press **CTRL-x**, select **Y** to save the file, then **enter** to confirm the filename. Note, the command rm removes files and directories. The command below is deleting a specific file in your home directory. **DO NOT ADD SPACES in the file path**.

user@host:~$ ls ~

user@host:~$ ls ~ >> ~/lab1-results.txt

user@host:~$ cat ~/new-file.txt

user@host:~$ cat ~/new-file.txt >> ~/lab1-results.txt

user@host:~$ rm ~/new-file.txt

user@host:~$ ls -a ~

user@host:~$ ls -a ~ >> ~/lab1-results.txt

* 1. Copying and Moving Directories and Files

There is no GUI, so there is no cut and paste when copying files. Instead, use the command **cp** and **mv**. Command **cp** is copy, **mv** is move. Either individual files or entire directories can be copied or moved. A set of files with common features, such as the same file extension or text string, can also be copied or moved using wildcards, and files can be renamed in the process.

user@host:~$ echo >> ~/lab1-results.txt

user@host:~$ echo Copy/Move Files/Dir >> ~/lab1-results.txt

user@host:~$ cp ~/lab1-results.txt ~/newdir

user@host:~$ cp ~/lab1-results.txt ~/newdir/lab1-cp.txt

user@host:~$ ls ~/newdir

user@host:~$ ls ~/newdir >> ~/lab1-results.txt

user@host:~$ mv ~/newdir/lab1-cp.txt ~/.hiddendir

user@host:~$ ls ~/newdir

user@host:~$ ls ~/.hiddendir

user@host:~$ ls ~/.hiddendir >> ~/lab1-results.txt

The following steps demonstrate **rm**. First, you will attempt to remove a directory that contains files. Note the results. Then you will use the –r option to recursively remove the directory and its contents. Last, you will delete files using a wildcard.

user@host:~$ rm ~/.hiddendir

user@host:~$ rm –r ~/.hiddendir

user@host:~$ rm ~/newdir/\*.txt

user@host:~$ ls ~/newdir

user@host:~$ ls –a ~

Directory newdir should exist, but be empty. Directory .hiddendir should be deleted.

* 1. **Installing Packages (software)**

There are two main methods to install software in Linux. The first is to compile and install the software from binary files. There are times when this is either required or preferred. However, compiling and installing software manually also puts the burden on the user to identify and install dependencies. A simpler and more common way to install software is through packages. Common tools used to install packages include **apt-get** (Debian-based distributions, including Ubuntu), **yum** (Fedora, Red Hat) and **urpm** (Mandriva). Another utility, **dpkg**, is used to manage packages in Debian-based distributions. This section introduces **apt-get** and **dpkg**. You will use **apt-get** to install mysql server, then **dpkg** to list installed packages. **dpkg** can also be used to reconfigure previously installed packages and other administrative tasks. Queries using **dpkg** are often long and complex. If you are interested in better understanding **dpkg**, query **dpkg** **--help** to understand the options in the commands shown below.

user@host:~$ echo >> ~/lab1-results.txt

user@host:~$ echo Install Packages >> ~/lab1-results.txt

**root** **is required for the following tasks.**

You will be prompted to create a root password for MySQL server during the installation process. You will need that password later in the semester, so ensure you know the MySQL server root password.

The **-y** option installs packages without being requested to confirm that you really want to install the packages. It can often be used, but there may be times where it is not desirable.

root@host:~# apt-get install mysql-server –y

If you haven’t figured it out yet, you can cut & paste commands into PuTTY. Select the command (the content after the prompt) and paste it into PuTTY by right-clicking inside the PuTTY terminal.

root@host:~# dpkg --get-selections | awk '/php/{print $1}' | xargs dpkg-query --show $1

The previous query displays a very long list of installed packages. The following command filters that output for just the mysql packages, and the third dpkg command writes the output to lab1-results.txt.

root@host:~# dpkg --get-selections | awk '/php/{print $1}' | xargs dpkg-query --show $1 | grep mysql

root@host:~# dpkg --get-selections | awk '/php/{print $1}' | xargs dpkg-query --show $1 | grep mysql >> /home/your-username/lab1-results.txt

root@host:~# exit

* 1. **Transfer files using SCP/PSCP**

You will need to transfer files to and from your JagCloud instance. If you had physical access to the host the instance was running on, you might be able to connect removable storage media to the host and transfer files to/from that removable media. However, you don’t have physical access, so you will need to use remote file transfer. SCP and pscp perform file transfer over ssh. Other options exist, such as netcat, ftp, or sftp, but since you will routinely access JagCloud instances with ssh, scp and pscp are good options for routine file transfer.

1. Start pscp.exe on your local computer. Search your computer for pscp.exe and start the executable. If you do not find pscp.exe on your computer, download pscp.exe from [here](https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html).
2. Start Windows Command Prompt by searching for **cmd**. Confirm pscp is functioning, then transfer the file. The following commands demonstrate how to complete these steps. **Note: the following commands are run through the Windows Command Prompt**, not the JagCloud instance.

pscp use is: pscp -i private-key username@server-ip-address:source destination

**-i**: -i option directs use of private key for authentication

**private-key**: The file path to the private key

**username**: The username on the server, the username on JagCloud instance

**server-ip-address**: IP address for JagCloud instance

**source**: The source file. Can be just the filename if the source file is in the user’s home directory, otherwise it must be the complete file path.

C:\Users\jdoe>**pscp –i C:\Users\jdoe\jdoe-private-key.ppk jane-doe@50.60.70.80:lab1-results.txt lab1-results.txt**

The above command will copy file lab1-results.txt from the home directory of jane-doe on the Linux server at IP address 50.60.70.80 to the directory C:\Users\jdoe on the local Windows computer. Alternatively, you can save the file to another directory by specifying the full file path for the desired directory.

Confirm the file copied to your local computer with the command:

C:\Users\jdoe>dir lab1-results.txt

**Submission:** Make two lab submissions.

1. Lab Submission
   1. Submit your lab notes
   2. Submit file lab1-results.txt