IT Club Cyber Defense Notes, 2022-11-10

In Attendance:	Anna, Chris, Dan, Michael, Tristan
Main Topics:	apt ssh
General info about apt	**apt** (short for aptitude) is a package manager for Ubuntu. It installs and maintains software packages on an Ubuntu system. It is one of two main package managers on new versions of Ubuntu; the other is called **snap**. **apt** uses a local database called a *repository* that tells it what software is available and which ones require each other in order to work.

Specific apt commands	**apt update** will update the local **apt** repository. It is important to run this command first whenever you want to use **apt**.
	
	`apt update manages the local apt repository.`
	
	`Other apt commands manage your software.`
	
	apt install installs new software packages.
	
	apt upgrade upgrades software packages that are already installed and for which upgrades are available.

Maintaining software with apt	Run these commands to update all programs that apt can update on your laptop.
	`apt update`
	`apt upgrade`

SSH	**ssh** is a program that allows you to run a terminal session on a remote computer. It does this by communicating with an ssh server on the remote computer. The ssh server is typically running a process called **sshd**, which stands for ssh daemon.

Installing ssh	To install the ssh server, run these commands.
	`apt update`
	`apt install openssh-server`

sshd configuration	Configuration settings for **sshd** are stored in a file named **sshd_config**. On current versions of Ubuntu, this file is located at **/etc/ssh/sshd_config**.
Key sshd configuration settings to learn right away	The following **sshd_config** configuration items are some of the more frequently-set ones. **AllowUsers**, **AllowGroups**: use these settings to allow SSH access for specific users. **DenyUsers, DenyGroups: use** these settings to disallow SSH access for specific users. *Note: if you do not use the above Allow or Deny settings for Users or Groups, the default policy of **sshd** allows all users to login via **ssh**.* **PermitRootLogin**: For now, always set this to **no**. If you need to take a remote action as root, there are other ways to do that.

Connecting to a remote computer using SSH	There are scads of **ssh** clients available. The most famous one is **PuTTY**. Feel free to love it. I hate it. Both Windows, Linux, and macos all have command-line ssh clients available that allow you to connect from a terminal window. The basic syntax to do this is **ssh userName@remoteComputer**. The remote computer can be specified either as a computer name or an IP address. In addition to **PuTTY** and command-line **ssh** programs, there are a lot of GUI-based **ssh** clients. For Windows, I use one from a company called **BitVise**. For Linux, I use one called **Remmina**.
Sample SSH session	Here's an example where I use the **hostname** command to show that I'm on a laptop named **LTTC-74HJCK3.** Then I use ssh to connect to another device, which in this case is a Wi-Fi router in my house. The device is a model called UniFi UAP-HD from a company called Ubuquiti, and all of this information is mentioned in the huge banner that is printed when I logon. I use a different command (**uname -n**) to display the hostname of the device (which is **hubble**) before running the **exit** command to end the remote session.

```
pwsh> hostname
LTTC-74HJCK3
pwsh> ssh skye-captain@hubble.area51.network
skye-captain@hubble.area51.network's password:


BusyBox v1.25.1 () built-in shell (ash)


  ___ ___       .__________.__
 |   |   |____  |__\_  ____/__|
 |   |   /    \ |  ||  __) |  |   (c) 2010-2022
 |   |  |   |  \|  ||  \   |  |   Ubiquiti Inc.
 |______|___|  /__||__/   |__|
            |_/                   https://www.ui.com

      Welcome to UniFi UAP-HD!

********************************* NOTICE **********************
* By logging in to, accessing, or using any Ubiquiti product, y
* signifying that you have read our Terms of Service (ToS) and
* License Agreement (EULA), understand their terms, and agree t
* fully bound to them. The use of SSH (Secure Shell) can potent
* harm Ubiquiti devices and result in lost access to them and t
* By proceeding, you acknowledge that the use of SSH to modify
* outside of their normal operational scope, or in any manner
* inconsistent with the ToS or EULA, will permanently and irrev
* void any applicable warranty.
***************************************************************

hubble-BZ.6.2.41# uname -n
hubble
hubble-BZ.6.2.41# exit
Connection to hubble.area51.network closed.
pwsh>
```

systemctl	We also discussed the **systemctl** command a bit. This command can be used to start a service, stop a server, or to check the status of a service. For example:

```
root@shellsburg:/var/log# systemctl status sshd
● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor pres
     Active: active (running) since Sat 2022-11-12 22:59:14 CST; 58s ago
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 12176 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/S
   Main PID: 12177 (sshd)
      Tasks: 1 (limit: 4626)
     Memory: 1.7M
        CPU: 12ms
     CGroup: /system.slice/ssh.service
             └─12177 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 start

Nov 12 22:59:14 shellsburg systemd[1]: Starting OpenBSD Secure Shell serve
Nov 12 22:59:14 shellsburg sshd[12177]: Server listening on 0.0.0.0 port 2
Nov 12 22:59:14 shellsburg sshd[12177]: Server listening on :: port 22.
Nov 12 22:59:14 shellsburg systemd[1]: Started OpenBSD Secure Shell server
root@shellsburg:/var/log# systemctl stop sshd
root@shellsburg:/var/log# systemctl start sshd
root@shellsburg:/var/log#
```

Homework

If you have time, try to do these things with SSH. You should have done some or most of these at Thursday's meeting. If so, just review what you've already done.

- Install the **openssh-server** using **apt**
  - Note: **openssh-server** installs the **ssh** client as well as the **sshd** server
- Use the **ssh** client to connect to your **Ubuntu** VM from a terminal on that **Ubuntu** VM.
- In the VirtualBox configuration settings for your VM, go to the network settings. In the field labeled **Attached to**, make sure it's set to **Bridged Adapter**.
  - If it's not, change the setting to **Bridged Adapter** and then reboot your VM.
  - You should reboot your VM just like you would reboot a real laptop.
    - For Ubuntu, clicking in the upper-right corner of the screen (where the Network,

	Battery, and Speaker icons are) will display a menu that has **Power Off / Log Out** at the bottom. Click that option and you should find an option to reboot the computer (the VM). ▪ Alternatively, if you are in a terminal window, you can run the command **reboot**. ○ After starting with Bridged Networking, your VM should have an address that your Windows laptop can "see" because it will be on the same subnet. ○ See if you can connect to your VM using **ssh** from your Windows OS. You can use command-line **ssh**, the **Bitvise ssh** client that I like, or any other **ssh** program you find. Hint #1: You might want to use your VMs IP address instead of its name in your ssh command. To find it, you can run the command **ip addr** in a terminal on your VM. Ask the group in Slack if you need help figuring out which line of output has the address you are looking for.
Extra credit	I know that some of you (most? all?) got the homework done at the meeting Thursday. If you'd like to try to go a little further with **ssh**, here's an idea. It is possible to log on to a remote computer securely, but without a password, using something called public/private keypairs. Do you think you can make this work? This is an intermediate topic, not a beginner's topic. Helpful hint: if you try to do this, you can create your keypair using the **ssh-keygen** program in Linux.