CASP+ Linux Systemd LAB

<u>Objective:</u>

1. TCP Based Service Turned Off and Prevented from Starting on an OS Reboot

<u>Initial Conditions:</u>

1. Ubuntu 15.04 or newer is installed and updated (this is when systemd was included by default in Ubuntu)
2. Apache2 server is installed:
   a. **sudo apt install apache2**
3. Net-tools are installed (provides netstat):
   a. **sudo apt install net-tools**
4. Verify Services Running:
   a. **systemctl status apache2**
   Should see apache2.service
   Loaded: ~…. enabled;
   Active: active (running)
5. Determine the IP Address of the Apache2 Server:
   a. **ifconfig**
6. Open Firefox and go to **http://[*URL from ifconfig*]**
   a. You should see the Apache2 Ubuntu Default Page

<u>Before we get started Q&A:</u>

Q: What is the difference between a Service and a Process?

A: A daemon is a long-running background process that answers requests for services. A service is a client/server application that also runs in the background; like apache2. A process is an application or script which can be running in the foreground or background; like firefox

Q: What is systemd?

A: A software suite that provides and array of components for the OS. It is run as the first process to boot (PID 1). Systemd has several core components and libraries, one of which is systemctl. Systemctl is used to inspect and control the state of the systemd system and the service manager. *Systemd tracks processes using the Linux Kernel's cgroups subsystem instead of using PIDs. This means resources can be managed per application rather than by the individual processes that make up an application.* This prevents daemons from double forking (parent, child, grandchild processes) their PIDs to escape shutdown.

Systemd has several Ancillary components, such as journald, localed, logind, networkd, resolved, system-boot, timedated, timesycd, tmpfiles, and udevd.

Q: Is systemd the only way to start / stop services?

A: No, there is also a 'service' family of commands and deprecated 'init scripts' that are still available in Ubuntu.

Q: What is the difference between a service being started and enabled or stopped and disabled?

A: Service Start/Stop affects the 'Active' status of the service. It is either 'inactive (dead)' or 'active (running). This represents the current state of the service. Service Enable/Disable affect the Loaded status of the service. It is either '….enabled;' or '…disabled'. This represents whether the service will be loaded to start when the OS boots up.

Q: If systemd doesn't use PIDs, how can I tie a running service to a protocol that it is using without first knowing the service that I am looking for?

A: netstat, along with the switches **-n, -a, -t, -l**

-n	shows connections in numeric format
-a	shows all connection types
-t	shows tcp connections
-l	shows services that are listening
-p	shows the associated program

Q: How can I read more about systemd and netstat without being overloaded with someone's opinion online?

A: In a Linux CLI: **man systemd** AND **man netstat**

Q: I am familiar with killing a process or service. In systemd, what is the difference between killing a service and stopping it?

A: First, understand that there is a Kill command outside of systemd (kill <PID>) and there is a systemd kill command (which uses systemctl). They are not the same. The kill command outside of systemd only kills the PID that is identified. Even if you kill the main process, it may or may not pull down the associated child processes (double forked processes can exacerbate this issue). However, within systemd, the **systemctl kill** command directly sends a signal to every process in the group (remember that systemd uses cgroups, not PIDs). So, it may meet your intent; however, the formal way of doing this is using the **systemctl stop** command, which goes through the official configured way to shut down a service, (i.e. it invokes the stop command configured with ExecStop= in the service file). **Systemctl stop** should be sufficient in most cases. **Systemctl kill** is the tougher version, for cases where you either don't want the official shutdown command of a service to run, or when the service is hung in other ways

Q: When configuring a service, sometimes my changes don't immediately show up, why?

A: In general, whenever you make a configuration change to a service in Linux, you have to restart the service. A configuration change itself does not modify the running service. Forgetting to restart the service can lead to confusion. In this lab, this will not be an issue, but it should always be kept in mind when working with service configurations.

<u>**Identify the Service that is running a TCP connection:**</u>

1. Explore the different outputs of: **netstat, netstat -t, netstat -nt, netstat -nat, netstat -natl**
2. You will notice that there are TCP and TCP6 connections:
    a. The tcp ESTABLISHED entry is the result of connecting a socket to the listener (active connection). The tcp6 LISTEN entry is for the passive connection at the listener socket. It

shows up as tcp6 , since it is spawned from a tcp6 listener; however it represents a connection from an IPv4 address. We have identified services on the host that are using TCP connections. Now we need to enumerate them.

3. Use the **-p** switch to determine the program that is activating the listener socket: **netstat – natpl**
    - a. Note: your results may differ with the privilege of the account you are using. To get the maximum results use: **sudo netstat -natpl**
4. We have not identified two services (programs) that are creating TCP connections: cupsd (which is a printer protocol) and apache2 (which is an http webserver).
    - a. We will assume that we identify the apache2 server as the vulnerability that we want to shut down and keep from restarting on a system reboot
5. Let's explore the configuration of apache2 with systemd:
    - **a. systemctl status apache2**
        - i. Note the Loaded and Active lines
        - ii. Apache2 is enabled (will restart on OS reboot) and is running
    - **b. sudo systemctl stop apache2**
    - **c. systemctl status apache2**
        - i. Apache2 is inactive (dead), but still enabled
        - ii. Reload the webpage on Firefox and note that the server is no longer running
    - d. Reboot Ubuntu
        - **i. systemctl status apache2**
            1. Apache2 is active and enabled
        - **ii. sudo systemctl stop apache2**
        - **iii. sudo systemctl disable apache2**
            1. Executing: /lib/system/system-sysv-install disable apache2
               Removed: /etc/system/system/multi-user.target.wants/apache2.service
            2. So, disabling removes /etc/systemd/system/multi-user.target.wants/apache2.service
            3. This is why the service does not restart on OS reboot
        - **iv. systemctl status apache2**
            1. Apache2 is inactive and disabled
    - e. Reboot Ubunti
        - **i. systemctl status apache2**
            1. Apache2 remains inactive and disabled
6. Bring back apache2 to a running status
    - a. Try step 5 with the **sudo sytemctl kill apache2** command
        - i. Observe that there is no difference in the outcome in this case; however, as discussed earlier, kill is a more immediate and non-standard way of ending a program and can be used if the service is hung.
7. Disabling the service still allows it to be started. If you want to prevent the service from being startable use:
    - **a. sudo systemctl mask apache2**
        - i. This completely disables a service by linking it to /dev/null; you cannot start the service manually or enable the service.
    - **b. sudo systemctl unmask apache2**

            i. This removes the link to /dev/null and restores the ability to enable and or manually start the service.

8. Systemd Sidenotes
    a. The systemctl family of commands will accept apache2 and apache2.service
    b. To view the loaded active services with systemd
        i. **systemctl --type=service --state=active**
    c. To view the loaded active runing services with systemd
        i. **systemctl --type=service --state=running**
        ii. **systemctl | grep running | less**
    d. To disable a service manually: systemctl to disables a service by going to /etc/systemd/system/multi-user.target.wants and removing the .service script from the directory
        i. Go to the /etc/systemd/system/multi-user.target.wants/ folder and **ls | less** to see apache2.service is present
        ii. Disable apache2 and notice that apache2.service is removed from this folder
        iii. Enable apache2 so that you can manually remove it
            1. **sudo rm apache2.service**
            2. Since the script was not called, the status will still show as enabled, however, if you reboot, it will not load and the status will be corrected to be disabled
9. Let's explore the service command
    a. **sudo service --status-all | less**
        i. Services with a '+' are running. Services with a '–' are not running
    b. **sudo service apache2 status**
    c. **sudo service apache2 stop**
    d. **sudo service apache2 start**
    e. Since systemd is running on the OS, in order to enable or disable the service you would need to manually remove or add the /etc/systemd/system/multi-user.target.wants/apache2.service
10. Let's explore the deprecated init command
    a. **sudo /etc/init.d/apache2 status | less**
        i. **cat /etc/init.d/apache2 | less**
        ii. Go to the /etc/rc2.d/ folder: ls | less
            1. Services that will start up start with an 'S' and those that will not start with a 'K'
    b. **sudo /etc/init.d/apache2 stop**
    c. **sudo /etc/init.d/apache2 start**
    d. **sudo /etc/init.d/apached2 restart**
11. If you are interested in diving even further into the rabbit hole of how to enable and disable services:
    a. https://askubuntu.com/questions/19320/how-to-enable-or-disable-services