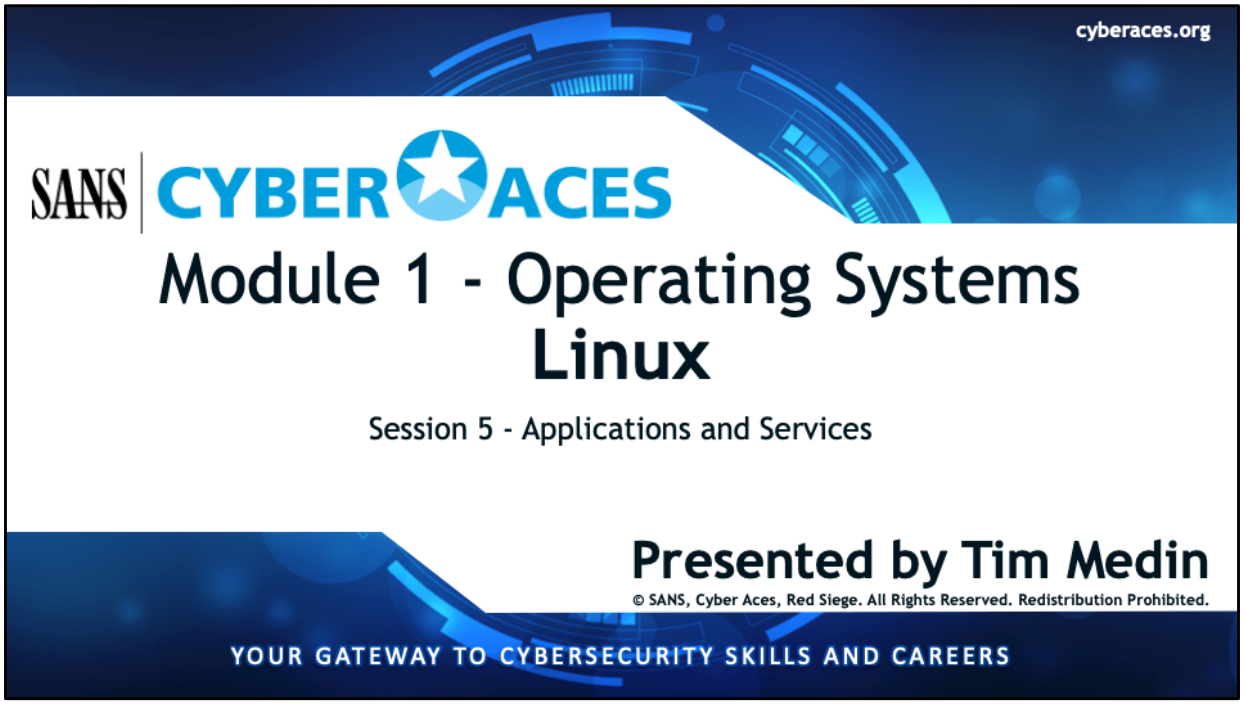
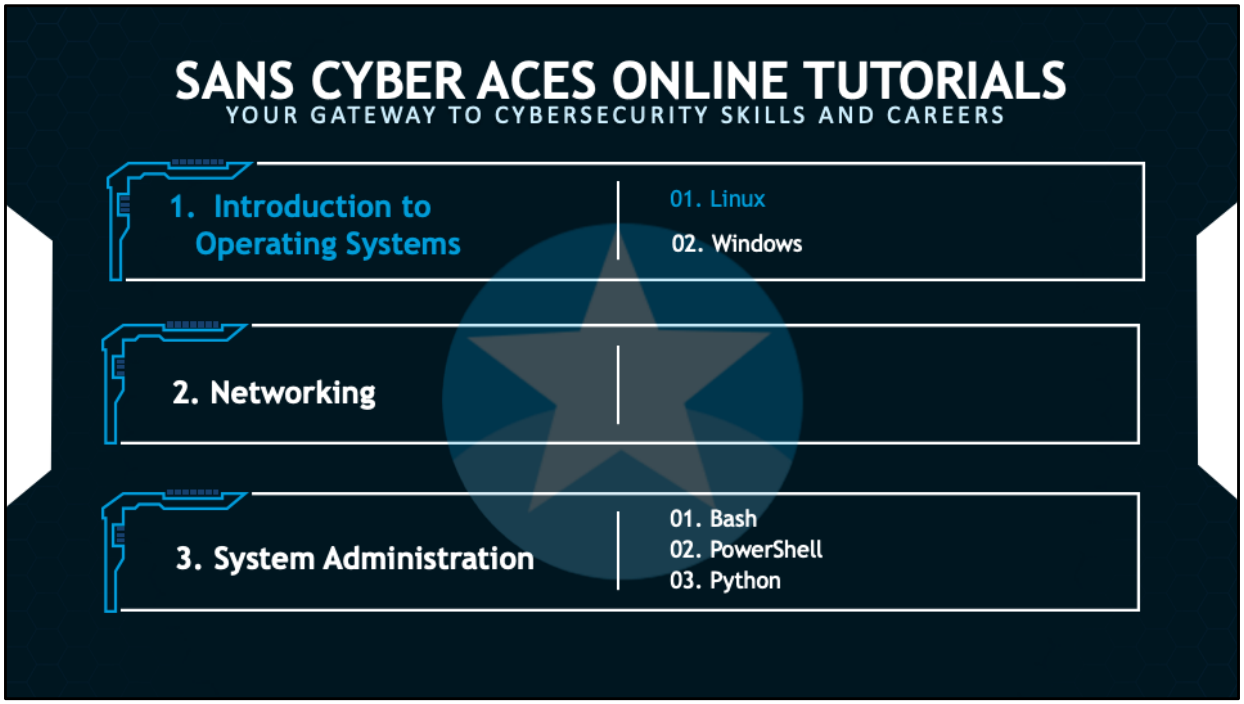


Welcome to Cyber Aces Online Module 1 - Linux. In this session we will discuss Linux applications and services.

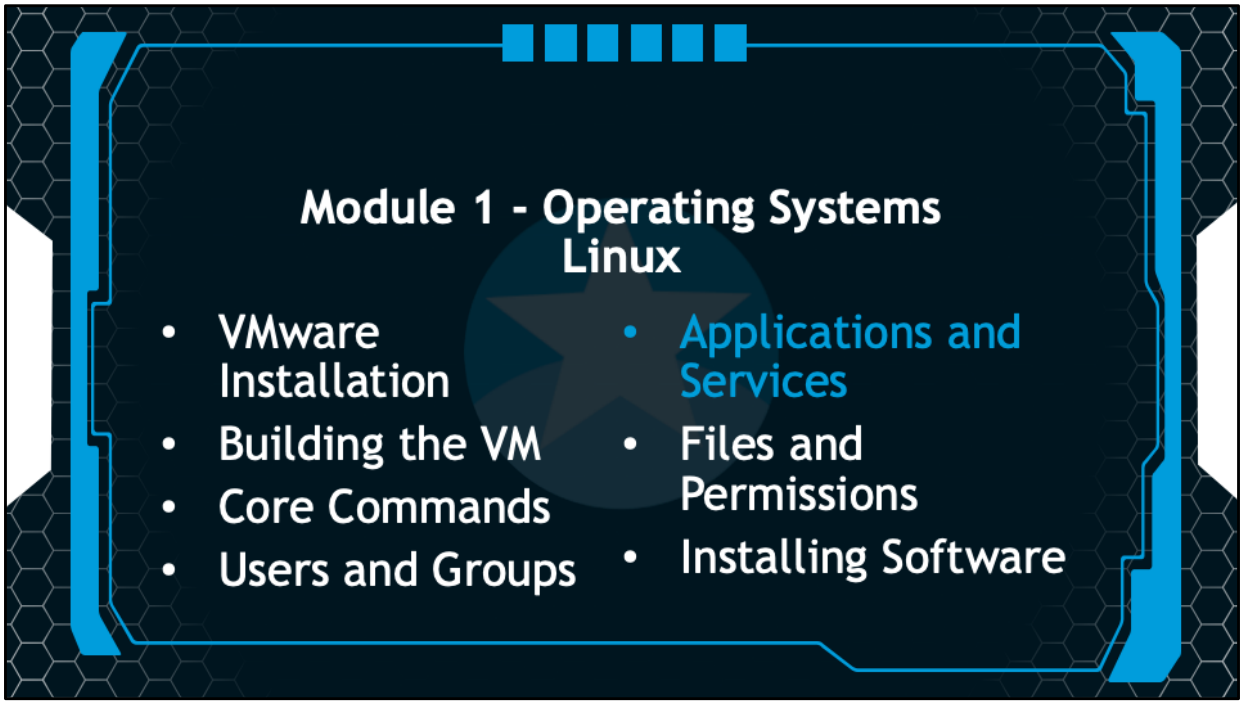
Content in this session has been developed by Tom Hessman, Tim Medin, Mark Baggett, Doug Burks, Michael Coppola, Russell Eubanks, Ed Skoudis, and Red Siege.



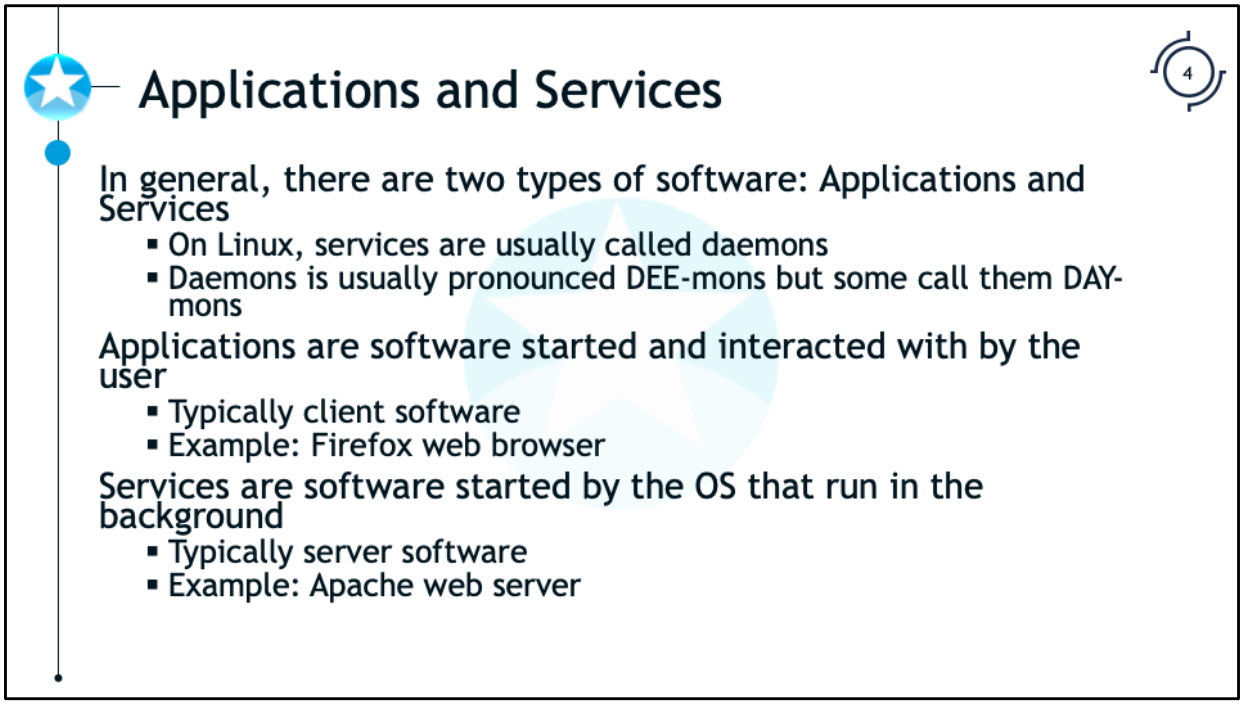
This training material was originally developed to help students, teachers, and mentors prepare for the Cyber Aces Online Competition. This module focuses on the basics of what an operating systems is as well as the two predominant OS's, Windows and Linux. This session is part of Module 1, Introduction to Operating Systems. This module is split into two sections, Linux and Windows. In this session, we will continue our examination of Linux.

The three modules of Cyber Aces Online are Operating Systems, Networking, and System Administration.

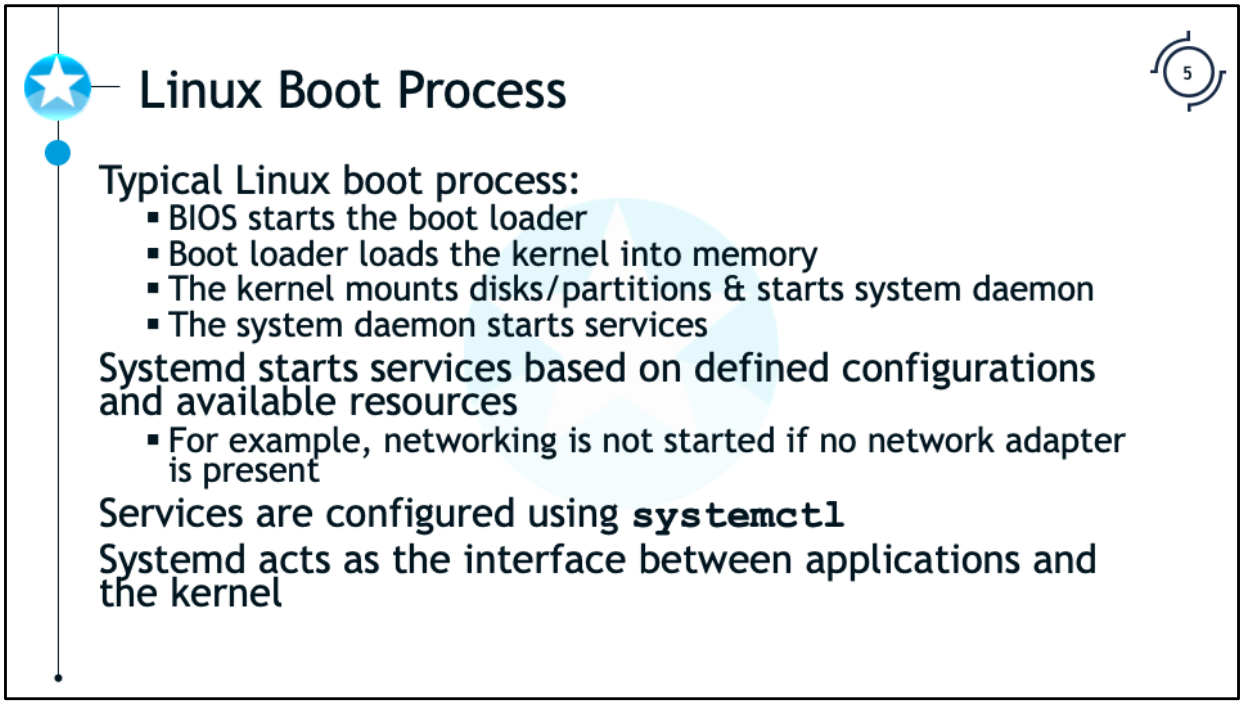
For more information about the Cyber Aces program, please visit the Cyber Aces website at https://CyberAces.org/.



In this session we will discuss Linux services, also know as daemons, and Linux applications.



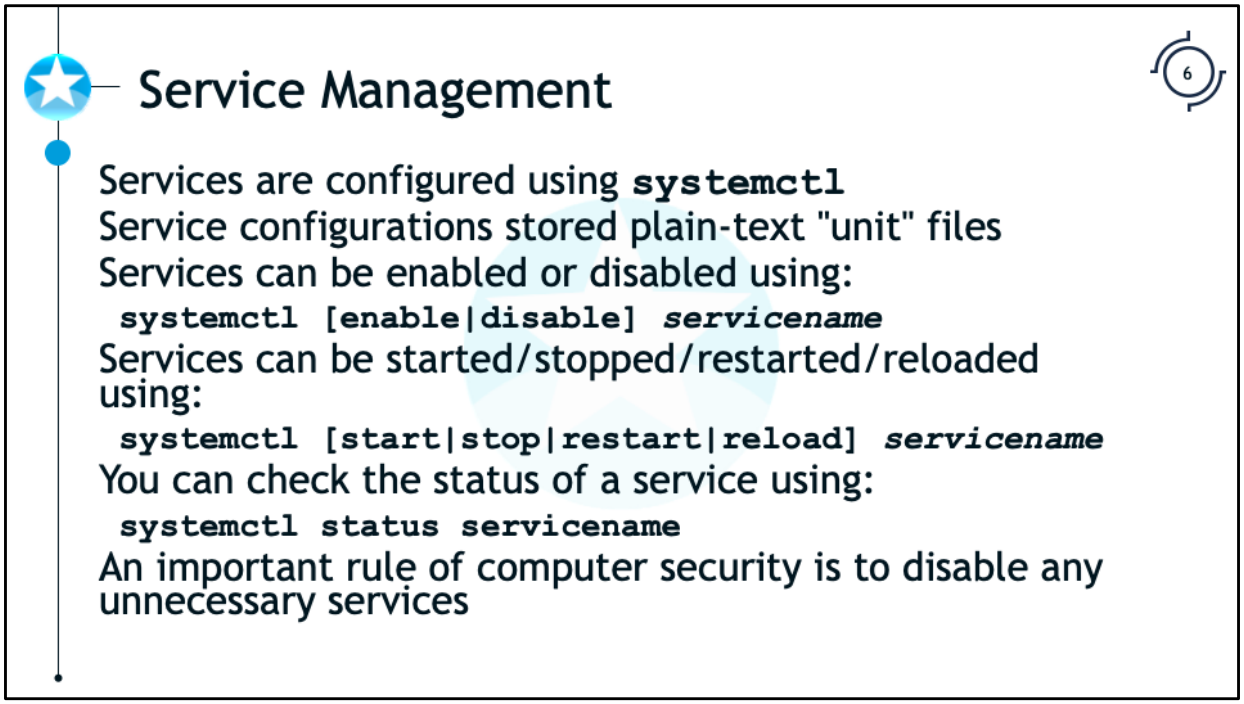
There are two categories of software that run on top of the operating system: applications and services (commonly called daemons on Linux). Applications are started by the user, interacted with, and then closed. For example, you sit down at your Linux VM, open your Firefox web browser, search for something on google.com, and then close your browser down. That's an application. Services, on the other hand, typically start when the computer boots up and run until the computer shuts down. An example of a service would be the HTTP service running on one of Google's servers that your web browser connects to. That HTTP service is started automatically by the operating system and will continue running until the operating system shuts down or an administrative user stops the service.



In a traditional Linux system, the boot process goes like this:

- The computer BIOS starts the boot loader, a program on the boot device (such as a hard drive or DVD)
- The boot loader loads the kernel into memory
- The kernel mounts the disk partitions needed to run the system and starts the init system, now known as systemd
- Systemd starts services based on pre-defined configurations and available resources. For example, if a network adapter is not present, networking-related services are not started

Historically, Linux started processes using the init system and pre-defined runlevels. In 2015, most major distros replaced the init system with system. Systemd replaced the traditional init daemon and the runlevel concept, in addition to the shell scripts used to launch services at boot time and shut down the system. In addition to starting services, systemd also manages service configurations and acts as the interface between applications and the kernel. You can read more about systemd here: https://redsiege.com/ca/systemd



How do we tell the operating system which services should be enabled at startup? We do that using systemctl. Systemctl is the management interface for systemd and can be used to start, stop, enable, disable, and configure services.

Service configurations are stored in unit files, typically stored in /etc/systemd/system. The operating system typically stores configuration files for default services in /lib/systemd/system, and a symlink to the appropriate file is placed in /etc/systemd/system. A good reference on unit files can be found at: https://redsiege.com/ca/units

Systemctl can be used to manage service execution. To enable or disable a service, use the "systemctl [enable|disable] servicename", for instance "systemctl enable sshd" to enable the sshd service. Similarly, systemctl is used to start, stop, restart, or reload a service. For example, use "systemctl start sshd" to start the sshd service. A good reference on the systemctl command can be found at: https://redsiege.com/ca/systemctl

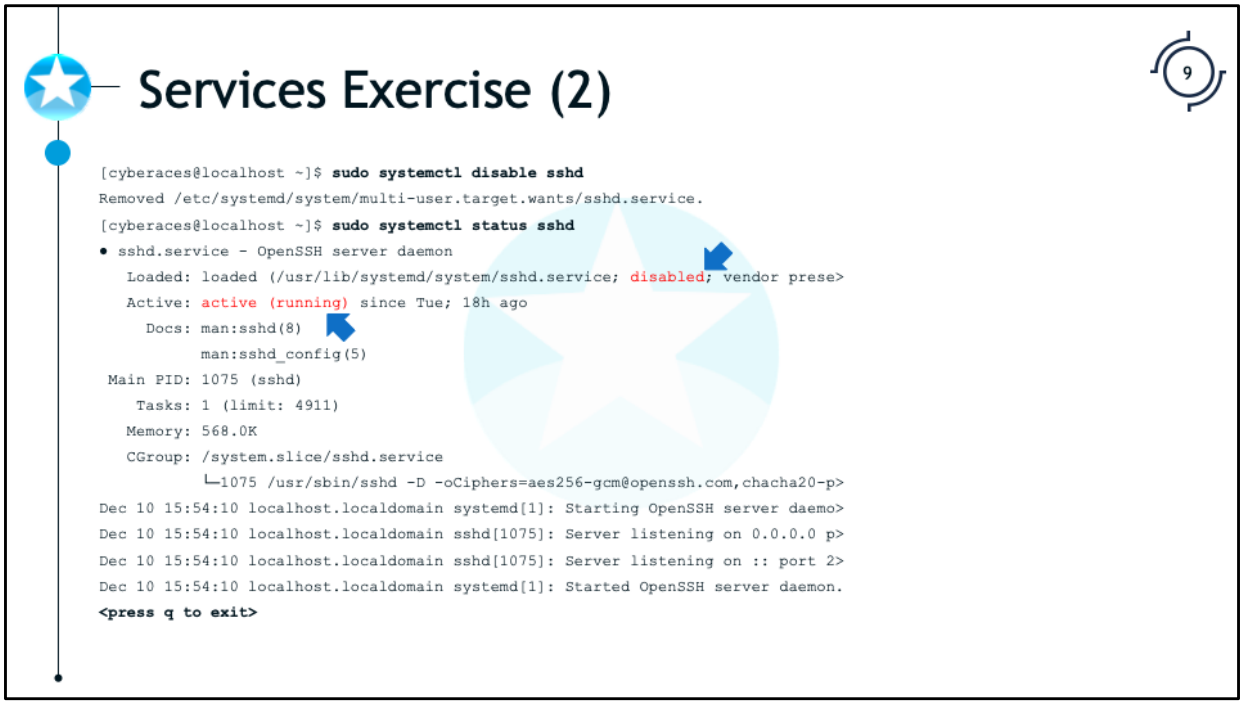
An important rule of computer security is to disable any unnecessary services.

The "ntsysv" tool runs at the command line, but provides an ncurses-based interface for managing services (ncurses is a library for creating semi-graphical applications at the CLI). To manage services, simply use the arrow keys to select a service, and use the space bar to enable or disable it. Then, press the tab key to move the cursor to the "Ok" button and press the space bar to "click" it.

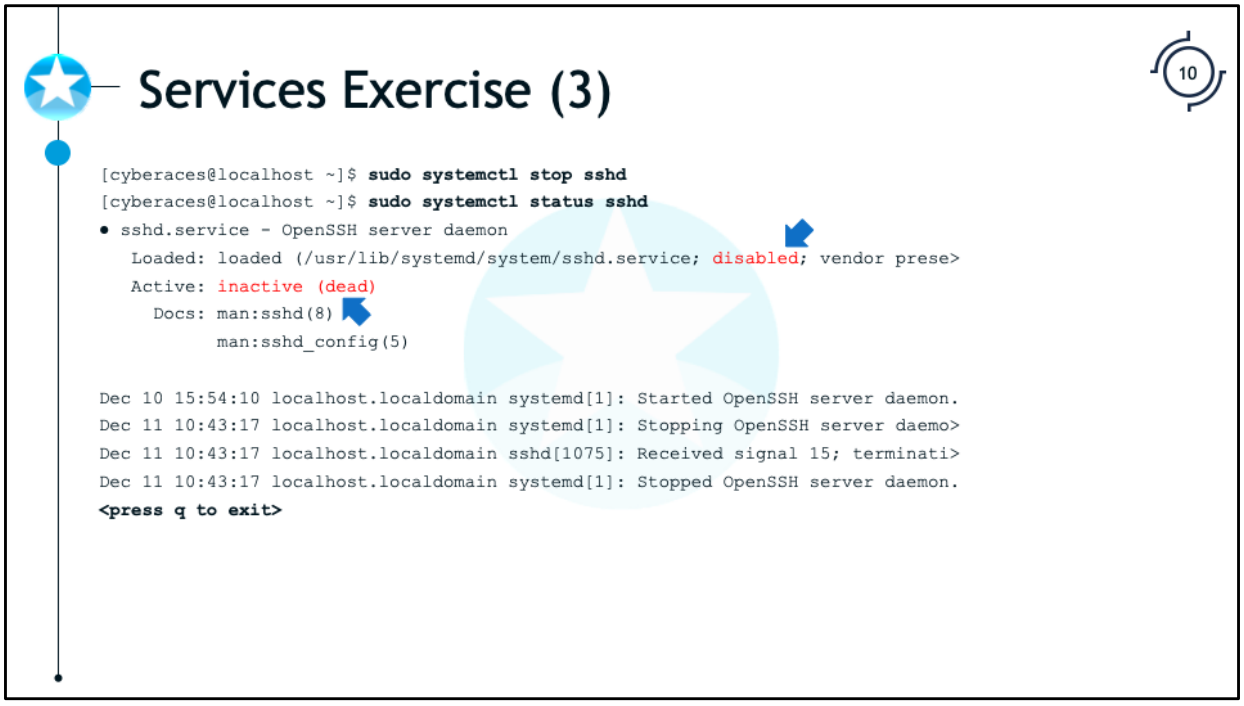
Services Exercise (1)


```
[cyberaces@localhost ~]$ sudo systemctl status sshd
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset>
   Active: active (running) since Tue; 18h ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 1075 (sshd)
    Tasks: 1 (limit: 4911)
   Memory: 568.0K
   CGroup: /system.slice/sshd.service
           └─1075 /usr/sbin/sshd -D -oCiphers=aes256-gcm@openssh.com,chacha20-p>
Dec 10 15:54:10 localhost.localdomain systemd[1]: Starting OpenSSH server daemo>
Dec 10 15:54:10 localhost.localdomain sshd[1075]: Server listening on 0.0.0.0 p>
Dec 10 15:54:10 localhost.localdomain sshd[1075]: Server listening on :: port 2>
Dec 10 15:54:10 localhost.localdomain systemd[1]: Started OpenSSH server daemon.
<press q to exit>
```

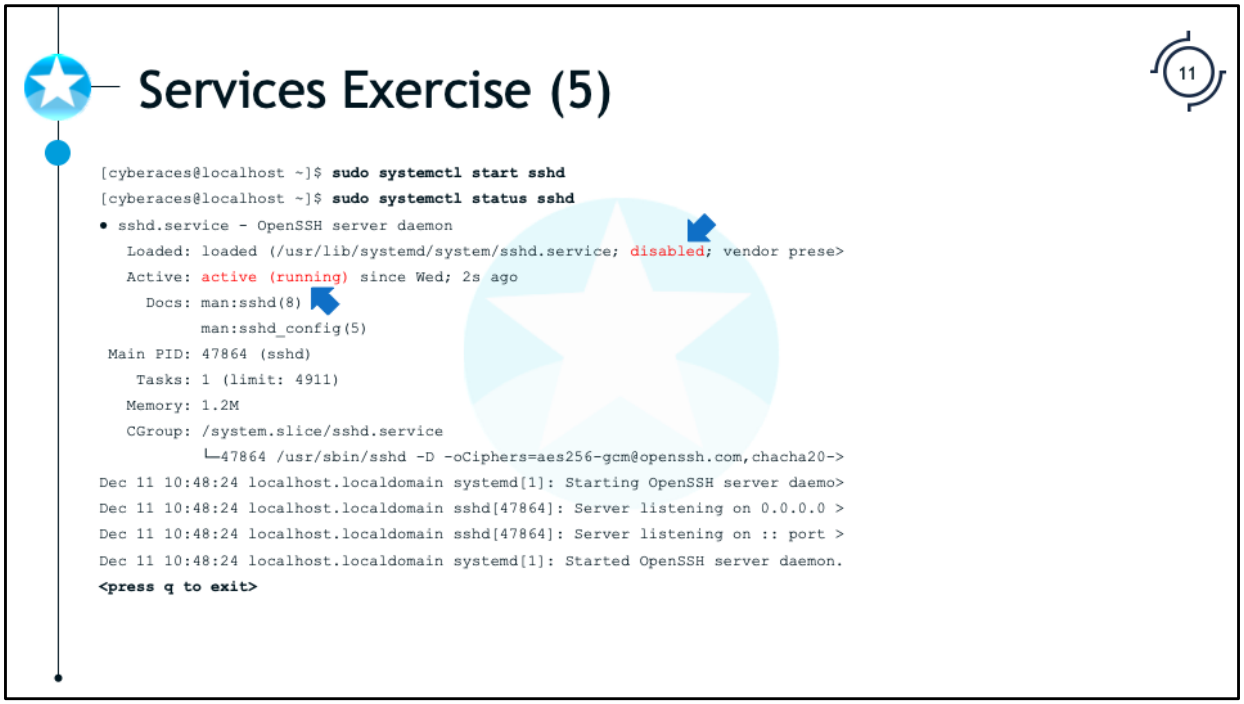
1. In your CentOS VM, open a terminal window and type "sudo systemctl status sshd" and hit enter. This returns the current status of the sshd service.
2. Note that the system reports the service is enabled and active. To exit the log press "q" to quit.



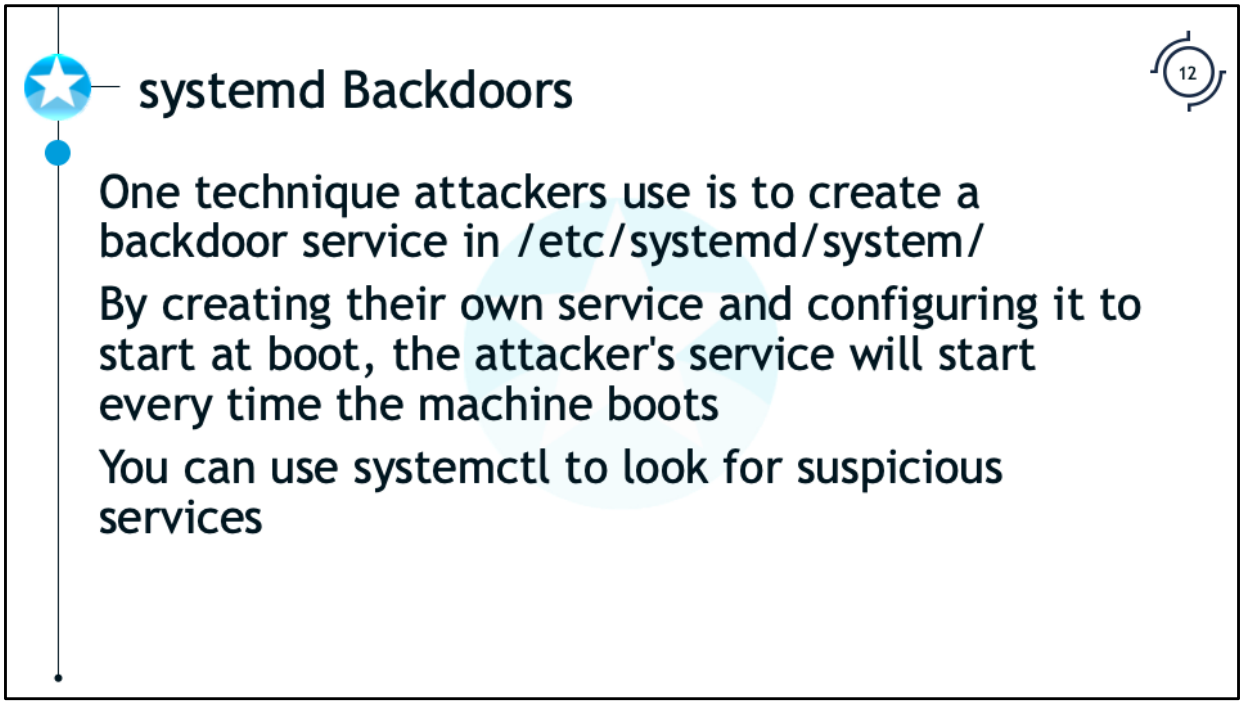
3. To disable the sshd service, in your terminal window, type "sudo systemctl disable sshd" and hit enter.
4. Examine the service again with "sudo systemctl status sshd" see that the service is disabled, but still running. To exit the log press "q" to quit.



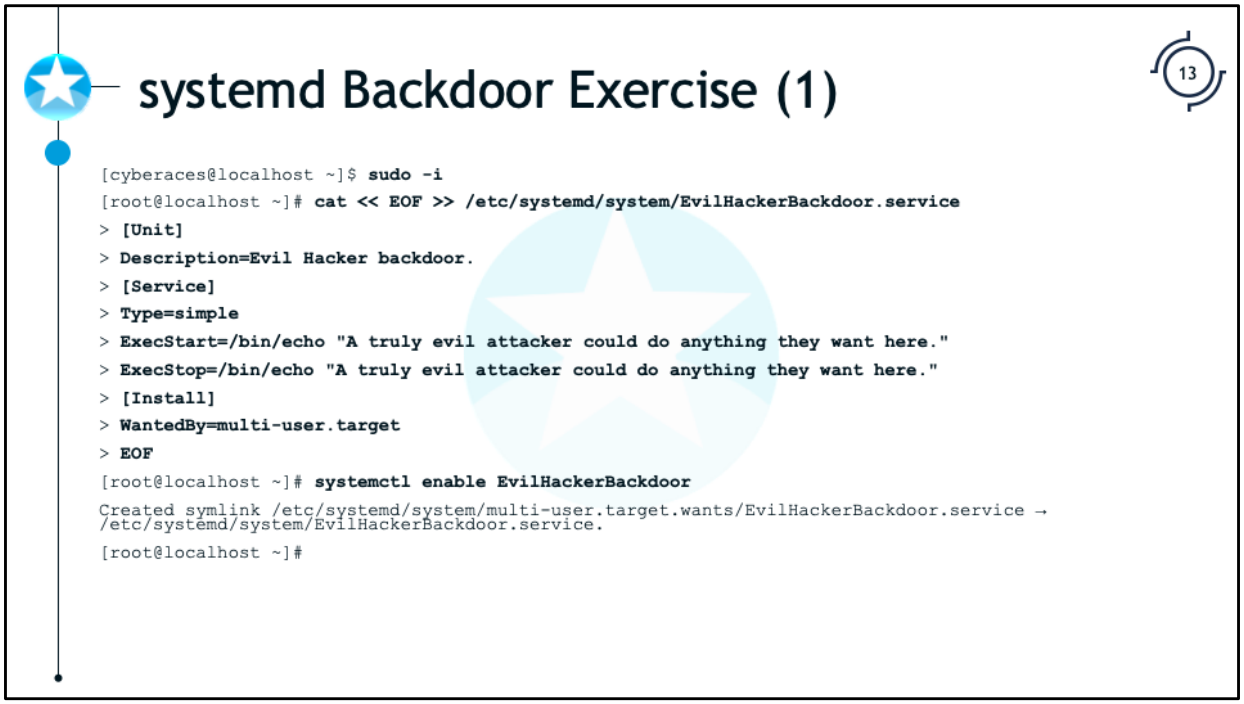
5. To stop the sshd service, in your terminal window, type "sudo systemctl stop sshd" and hit enter.
6. Examine the service again with "sudo systemctl status sshd" see that the service is disabled and inactive. To exit the log press "q" to quit.



7. To start the sshd service, in your terminal window, type "sudo systemctl start sshd" and hit enter.
8. Examine the service again with "sudo systemctl status sshd" see that the service is disabled and active. To exit the log press "q" to quit. Notice, that even if the service is disabled, it can still be started. The "disabled" means it will not be launched when the system is booted.



One technique used by computer attackers is to create a backdoor in /etc/systemd/system/. Recall that this is the directory that specifies system service units. If attackers can install their own service in this directory (by writing a service unit file) and enable the service with systemctl, then the attackers' service will start every time the machine boots and the bad guys will always have access to the box. You can use systemctl to find services that look suspicious.



1. In your CentOS VM, start a terminal (using either the panel or desktop icon you created earlier) and become root using the following command:

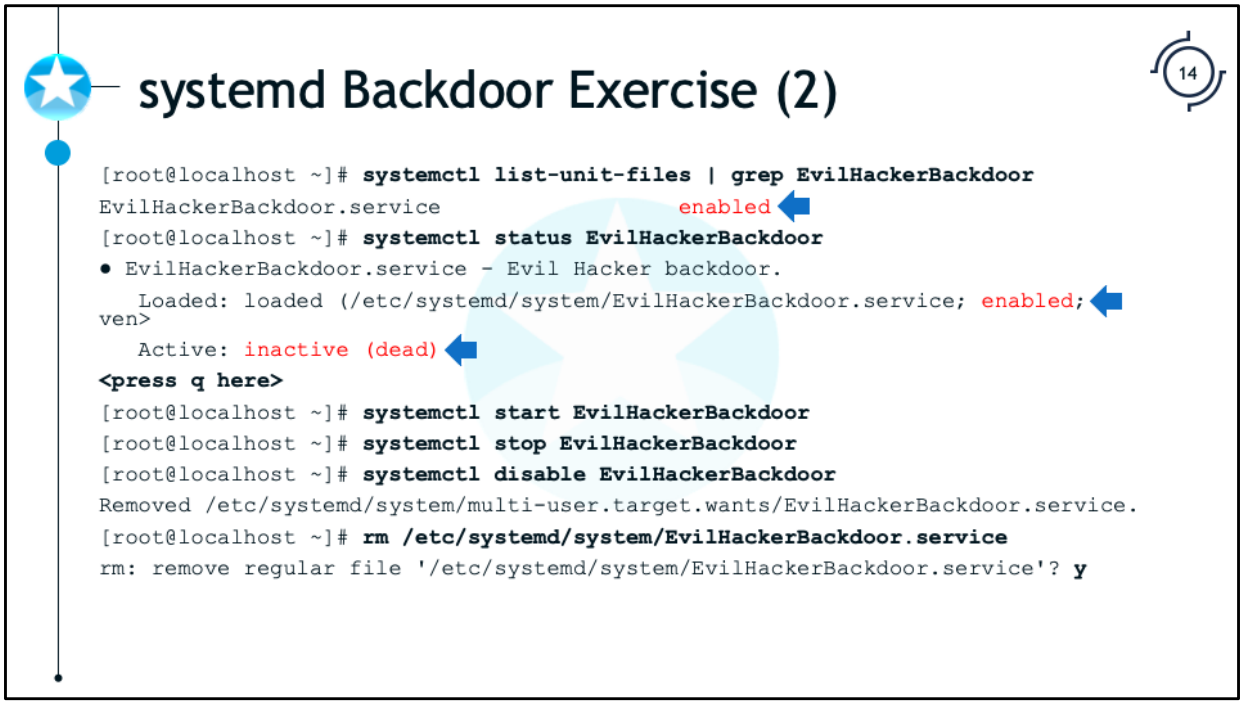
   $ **sudo -i**

2. Now that you are root, type the following command to create a systemd backdoor service in /etc/systemd/system/. Type everything below to create the script. This techniques uses a "heredoc" to create the script. Note, do not type the **>** shown above, as it is the prompt.

```
# cat << EOF >>
/etc/systemd/system/EvilHackerBackdoor.service
[Unit]
Description=Evil Hacker backdoor.
[Service]
Type=simple
ExecStart=/bin/echo "A truly evil attacker could do
   anything they want here."
ExecStop=/bin/echo "A truly evil attacker could do
   anything they want here."
[Install]
WantedBy=multi-user.target
EOF
```

3. Enable the backdoor using the following command:

```
# systemctl enable EvilHackerBackdoor
```



4. Verify backdoor was installed using list-unit-files (grep is used to search for a particular string of text):

   # **systemctl list-unit-files | grep EvilHackerBackdoor**

5. Verify the status of the backdoor service

   # **systemctl status EvilHackerBackdoor**

6. Start the backdoor:

   # **systemctl start EvilHackerBackdoor**

7. Stop the backdoor:

   # **systemctl stop EvilHackerBackdoor**

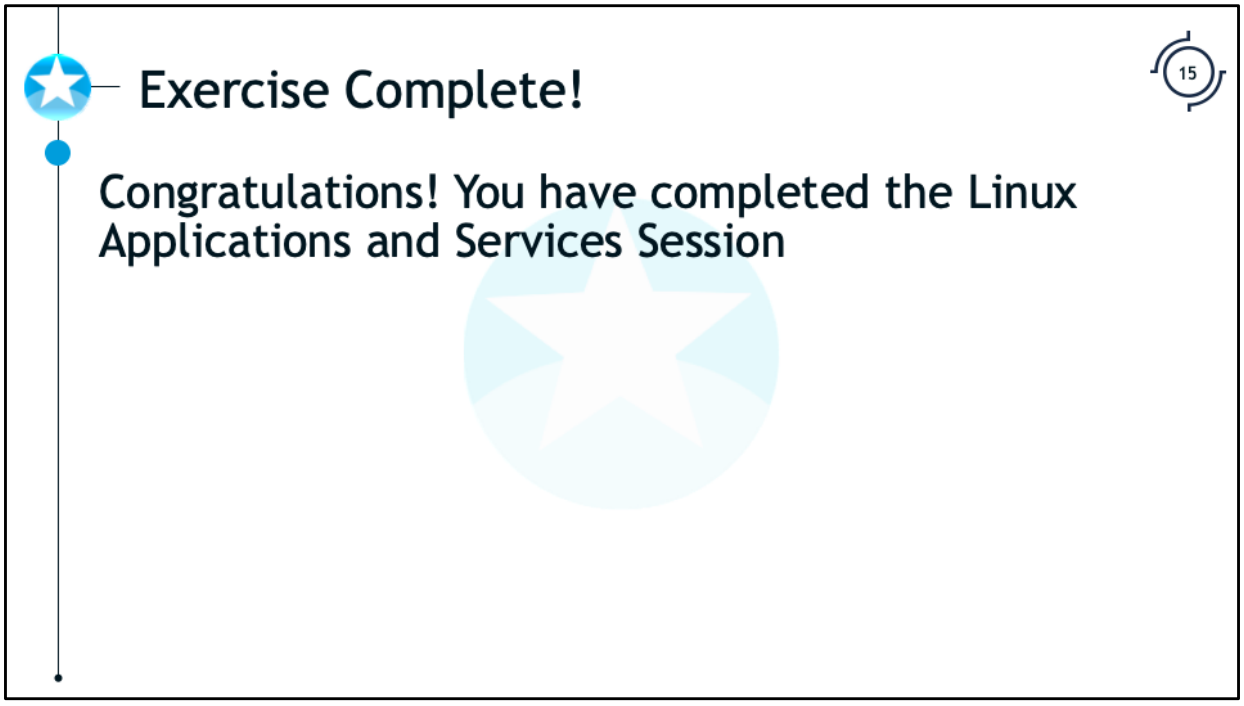
8. Disable the backdoor

   # **systemctl disable EvilHackerBackdoor**

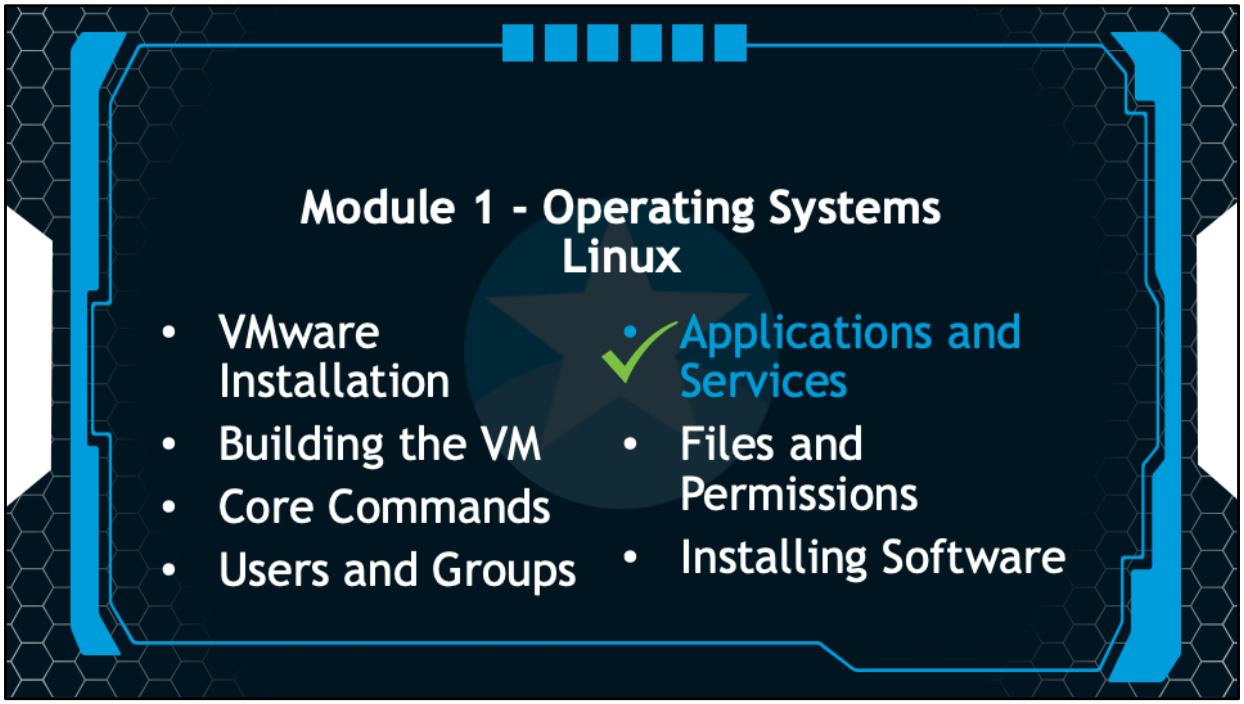
9. Remove the backdoor script (press "y" to confirm):

   # **rm /etc/systemd/system/EvilHackerBackdoor.service**

10. Close the terminal window.



Congratulations! You have completed the Linux Applications and Services Session.



You have completed the session on Linux applications and services. In the next session, we will discuss permissions and the file systems in Linux,