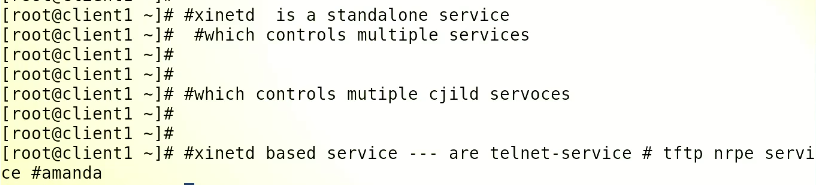
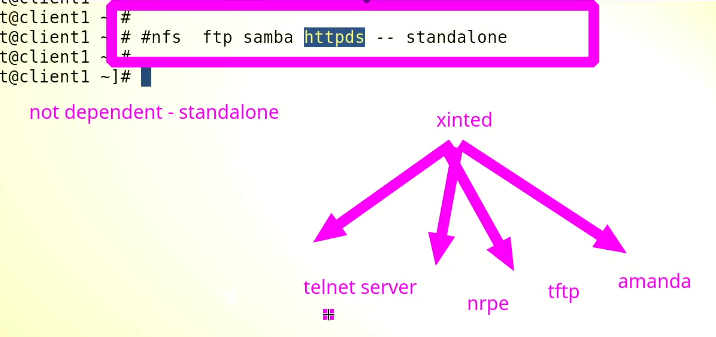
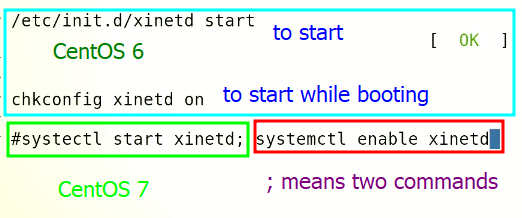
Lecture 18

**Xinetd-SSH**

**xinetd**



* The services we have learnt so far,
* nfs, httpd, ftp , samba à these are all services à there is not a single service which is dependent upon these services à but “xinetd” is a service which has child services.
* *these services work as a “daemon” à and run in background continuously until stopped.*
* As we know “daemon” is a service which run in background.
* ***Xinetd is also a standalone service à many child services inside “xinetd” which depend upon “xinetd”***
* ***.***
* 
* To start a specific dependent service
  + We will not use this command ***$ service <dependent\_service> start***
  + Instead, **$ service xinetd start** command will be used.
  + 
* .
* **xinetd**, short for extended Internet services daemon, is a powerful and flexible replacement for the traditional inetd daemon on Unix-like systems. It is a super-server that manages multiple network services on behalf of other servers or daemons, allowing them to be started on demand and terminated when not in use, conserving system resources.
* *xinetd acts as an intermediary between the system and the services it manages, and provides many features, including access control, rate limiting, logging, and the ability to launch services on demand or in response to incoming network traffic.*
* xinetd is commonly used to manage services such as ***FTP, telnet, ssh***, and others, and can be configured through a simple configuration file. It is widely used on Unix and Linux systems, and is often preferred over inetd due to its enhanced security features and flexibility.
* .
* Check if it is already installed and if not, install it
* 
* 

**service still managed by xinetd in CentOS 7**

While CentOS 7 excludes some services from being managed by xinetd, there are still several services that are managed by xinetd by default. These services include:

1. daytime-stream - This service provides the current date and time in a standardized format, for compatibility with older systems.
2. discard-stream - This service discards any data sent to it, for testing purposes.
3. chargen-stream - This service generates a stream of characters, for testing purposes.
4. echo-stream - This service sends back whatever it receives as a response, for testing purposes.
5. time-stream - This service provides the current time in a standardized format, for compatibility with older systems.
6. tcpmux-server - This service allows multiple TCP-based services to share the same IP address and port number.

These services are typically used for testing or compatibility purposes, rather than for production use. It is worth noting that xinetd can be configured to manage additional services, or to stop managing the default services if they are not needed. However, it is important to carefully consider the security implications of any changes to the xinetd configuration.

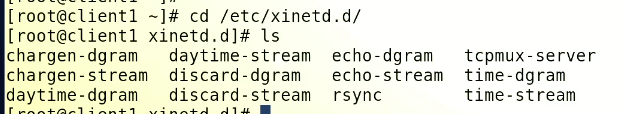
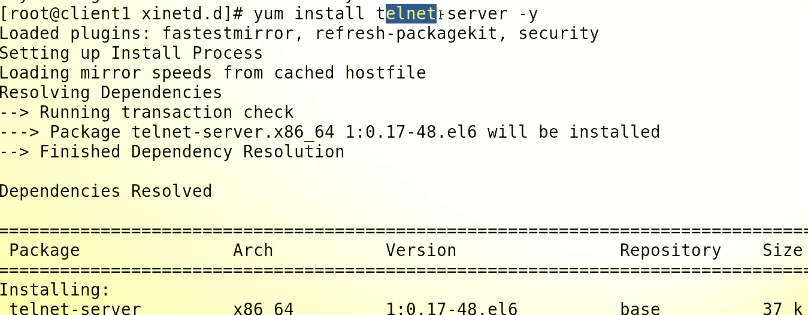
**services excluded from xinetd in centOS 7**

By default, CentOS 7 excludes some services from being managed by xinetd, the extended internet services daemon. This is done for security reasons, as some of these services may be vulnerable to attacks or may not be necessary for the normal functioning of the system.

The services excluded from xinetd in CentOS 7 include:

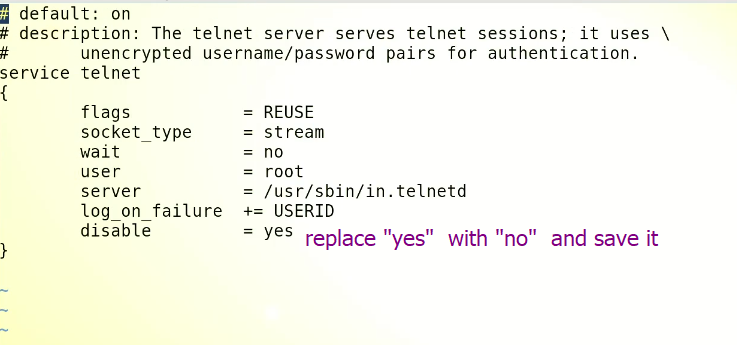
1. sshd - Secure Shell (SSH) is a secure protocol used for remote access to systems. It is excluded from xinetd as it is usually managed by systemd, the system and service manager.
2. telnet - Telnet is a protocol used for remote access to systems, but it is not secure and has been largely replaced by SSH. Telnet is excluded from xinetd for security reasons.
3. ftp - File Transfer Protocol (FTP) is used for file transfer between systems, but it is also not secure and has been largely replaced by more secure protocols such as SFTP or SCP. FTP is excluded from xinetd for security reasons.
4. daytime - The daytime service provides the current date and time on the system. This service is not necessary for normal system operation and is excluded from xinetd.
5. chargen - The chargen service generates a stream of characters for testing purposes. This service is not necessary for normal system operation and is excluded from xinetd.
6. echo - The echo service sends back whatever it receives as a response. This service is not necessary for normal system operation and is excluded from xinetd.
7. time - The time service provides the current time on the system. This service is not necessary for normal system operation and is excluded from xinetd.

It is worth noting that some of these services may still be enabled and running on the system, but they are not being managed by xinetd. For example, sshd is usually managed by systemd, while ftp and telnet may be managed by other daemons such as vsftpd or inetd.

* .
* ***Sir Kazim used “xinetd” on CentOS 6 for better understanding.***
* Root path
* 
* Inside this directory, it contains configuration file of all child service.
* By default these child services are disabled
* To install “telnet-server”
* 
* Now it is listed in the default path of xinetd
* Text

  Description automatically generated

This telnet is not conventional telnet which we used to check ports. $ telnet <IP> <Port\_No.>

* ***$ service xinetd start*** à it works in a way that if a specific service is required, it loads it into the RAM, and after finishing the job it clears the RAM.
* To enable telnet à open its configuration file vi telnet
* 
* ***$ service reload xinetd*** à to include “telnet” in running services while required.
* To check dependencies of an rpm package à ***yum deplist <package\_name>***
* .
* A picture containing diagram

  Description automatically generated
* Text

  Description automatically generated with low confidence

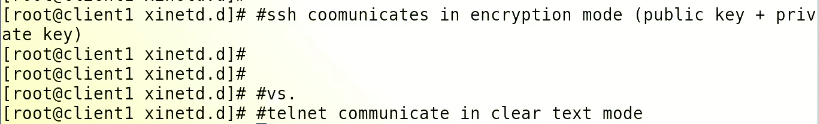
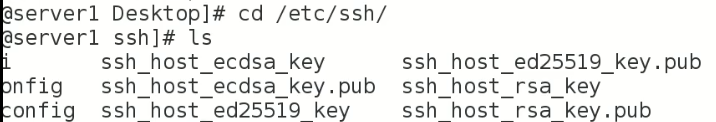
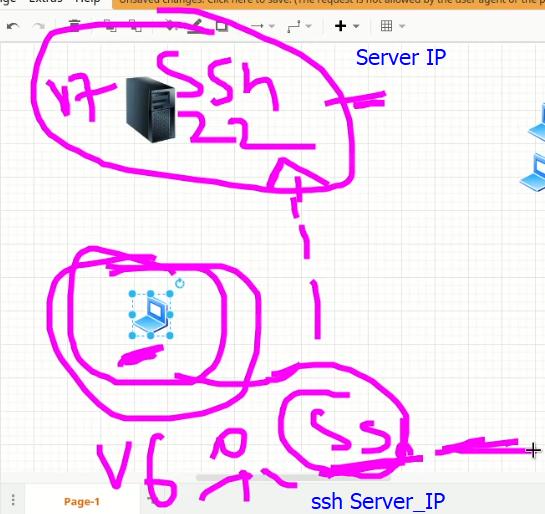
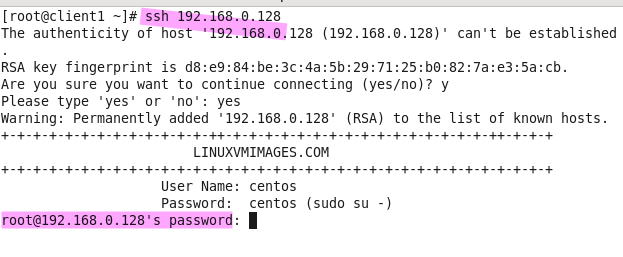
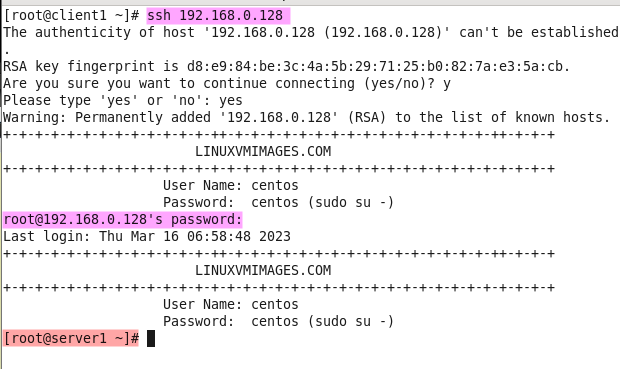
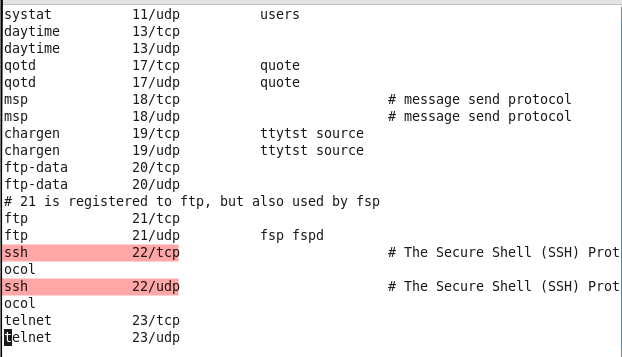
**SSH** (Secure Shell) à port No. 22 à used for remote login

SSH (Secure Shell) is a network protocol used for secure remote access to Linux and other Unix-like operating systems. It provides a secure, encrypted communication channel over an insecure network, such as the internet. With SSH, you can log in to a remote server from your local machine, execute commands remotely, and transfer files securely between machines.

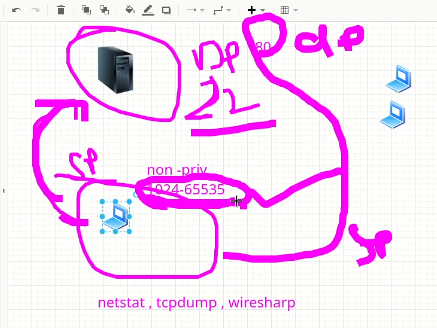
When you use SSH to connect to a remote server, you need to provide a username and password or a private key for authentication. SSH uses public-key cryptography to ensure secure communication between the client and server. This involves generating a pair of public and private keys, where the public key is shared with the remote server and the private key is kept on your local machine. When you log in to the remote server, SSH uses your private key to authenticate you without transmitting your password over the network.

SSH provides a command-line interface for executing commands on the remote server, as well as file transfer capabilities through the use of SCP (Secure Copy) and SFTP (Secure File Transfer Protocol). With SCP, you can copy files between your local machine and the remote server, while SFTP provides a more advanced file transfer capability with support for file permissions, directory structures, and other features.

In summary, SSH is a widely used and important tool for managing Linux and other Unix-like operating systems remotely. It provides secure authentication, encrypted communication, and advanced file transfer capabilities for managing servers and other remote machines.

* 
* .
* Remote session b/w server and client à the session receives to the client (encrypted).
* ***SSH uses Port No. 22 by default***
* SSH is a default package in mose Linux distros.
* *$ systemctl status ssh*
* $ telnet 127.0.0.1 22 à to check port
* $ netstat -tulnp | grep -I 22 à to check port
* ***Root directory***
* /etc/ssh/
* 
* How it works,
* .
* On client CentOS 6
* .
* .
* $ ssh <server\_IP> -p <Port\_N0> à to connect to a ***specific port à*** *destination port.* It will not go to “/etc/services” file
* 
* ***SSH*** *is just a command on client, even it is stopped, it will still work.*
* Connection established and “server1” is open.
* 
* **Tip:-** the machine that serves and its any port is open is called server (regardless the OS). If CentOS 7 connects CentOS 6 on port 8080, in this case CentOS 6 is now a server and vice vera.
* .
* *Client à reads à* ***/etc/services*** *file à it files tells the client that Port No. 22 is SSH port and you will be connected on it.*
* 
* If we change to Por No. in this file à 2222
* And try to connect $ ssh <server\_IP> 2222 à the **connection refused** error will display.
* **It is because Port No. 2222 is not opened on server1**
* It’s a troubleshooting step if a connection is not working à go to /etc/services and check the port No.
* ***To bypass /etc/services file use “-p” flag***
* To stop SSH à $ exit
* .

Next topic in SSH

* What would be the ***source port (the port on the client which send the SSH query out***) while establishing SSH connection? The port on server is “destination port” and the port on the client is “source port”.
* Source port is “***non privileged port”*** à **ranges b/w 1024 - 65535**
*  .
* =================
* /etc/ssh/sshd\_config à it is **SSH configuration file.**

**TASK**

* **Change the default Port No. 22 for SSH.**
* ***(port of any service can be changed by its configuration file)***

To change the default port number 22 for SSH in CentOS 7, you can follow these steps:

1. Login to your CentOS 7 server using SSH with your existing port 22.
2. Open the SSH configuration file **/etc/ssh/sshd\_config** using your preferred text editor. You can use the following command to open the file with the nano editor:

**sudo nano /etc/ssh/sshd\_config**

1. Locate the line that specifies the SSH port number, which should look like this:

#Port 22

Uncomment the line by removing the **#** symbol and change the port number to your desired port number. For example, to change the port number to 2222, modify the line to look like this:

Port 2222

1. Save the changes to the configuration file and exit the text editor.
2. Restart the SSH service for the changes to take effect by running the following command:

**sudo systemctl restart sshd**

1. After restarting the SSH service, you should be able to connect to your CentOS 7 server using the new port number. For example, to connect to the server using port 2222, you would use the following command:

**ssh username@server\_ip -**p 2222

Note: Before making any changes to your server's SSH configuration, ensure that you have another way to access your server in case you make a mistake or something goes wrong.

* If port is not up after these steps,

If you have followed the steps to change the SSH port on your server, but the port is not up after restarting the SSH service, there could be a few possible reasons:

1. Firewall rules: If you are using a firewall on your server, you may need to update the firewall rules to allow traffic on the new SSH port. Make sure that your firewall rules allow incoming traffic on the new port.
2. SELinux: If SELinux is enabled on your server, it may be blocking traffic on the new SSH port. You can use the **semanage** command to allow SSH traffic on the new port. For example, to allow SSH traffic on port 2222, you can use the following command:

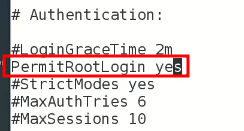
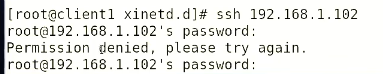
**sudo semanage port -a -t ssh\_port\_t -p tcp** 2222

1. Network configuration: Make sure that your server's network configuration is correct and that it is reachable from the outside on the new port. You can use the **netstat** command to check if the port is listening. For example, to check if port 2222 is listening, you can use the following command:

**sudo netstat -tulpn |** grep 2222

If none of these solutions work, you can try changing the port number again and repeat the steps. If the problem persists, you may need to consult with a system administrator or your hosting provider for further assistance.

In professional environment, the “root” privilege is not given to remote session through SSH, as per security requirement,

* To disable the root login for SSH,
* $ vi /etc/ssh/sshd\_config à
*  change it to “no”
*  à root cant be logged in from client now.
* To change root prompt à
  + $ ssh <IP> -l <user\_name\_at\_server>
  + $ ssh <user\_name\_at\_server>@<erver\_IP>
* If the “SSH” service is topped in server, the remote access will not work.
* For “password less authentication”
* ***Private key & Public key is generated. (TASK)***
* .
* **Extra reading for SSH**

***how to monitoring an incoming SSH request on server : CentOS 7***

|  |
| --- |
| To monitor incoming SSH requests on a CentOS 7 server, you can follow these steps:   1. Connect to your CentOS 7 server using SSH or any remote access method. 2. Install the "auditd" package by running the following command:   sudo yum install auditd   1. Once the installation is complete, start the auditd service and enable it to start at boot time by running the following commands:   sudo systemctl start auditd.service sudo systemctl enable auditd.service   1. To monitor SSH activity, you can create a rule in the auditd configuration file. Open the configuration file by running the following command:   sudo vi /etc/audit/audit.rules   1. Add the following rule to the configuration file:   -a always,exit -F arch=b64 -S execve -C uid!=euid -F euid=0 -F path=/usr/sbin/sshd -k sshd-exec  This rule will audit all executions of the "/usr/sbin/sshd" binary with root privileges.   1. Save and close the configuration file. 2. Restart the auditd service to apply the changes by running the following command:   sudo systemctl restart auditd.service   1. To view the SSH audit logs, you can use the "ausearch" command. For example, to view the SSH activity in the last hour, run the following command:   sudo ausearch -k sshd-exec -ts recent  This command will display a list of SSH events with timestamps, including the user who initiated the connection and the IP address they connected from.  That's it! You can now monitor incoming SSH requests on your CentOS 7 server. |

|  |
| --- |
| Task:- **Passwordless** login on a Linux through **ssh** |
| In this example, we will set up **SSH password-less** automatic login from server **192.168.0.12** as user **tecmint** to **192.168.0.11** with user **sheena**. Step 1: Create Authentication SSH-Keygen Keys on – (192.168.0.12) First login into server **192.168.0.12** with user **tecmint** and generate a pair of public keys using the following command.  **$ ssh-keygen -t rsa**    +-----------------+ Step 2: Upload SSH Key to – 192.168.0.11 Use SSH from server **192.168.0.12** and upload a newly generated public key (**id\_rsa.pub**) on server **192.168.0.11** under **sheena**‘s **.ssh** directory as a file name **authorized\_keys**.  **$ ssh-copy-id** [**sheena@192.168.0.11**](mailto:sheena@192.168.0.11)  Make sure that the correct permissions are set on the **~/.ssh** directory and the **~/.ssh/authorized\_keys** file on the remote server.  **$ ssh** [**sheena@192.168.0.11**](mailto:sheena@192.168.0.11) **"chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized\_keys"** Step 3: Disable Password Authentication (Optional) For increased security, you can disable password authentication on the remote server and only allow SSH key authentication. To do this, open the SSH server configuration file on the remote server:  **$ sudo nano /etc/ssh/sshd\_config** **OR** **$ sudo vi /etc/ssh/sshd\_config**  Find the line containing **PasswordAuthentication** and set it to **no**.  **PasswordAuthentication no**  Save the file and restart the SSH service.  **$ sudo systemctl restart sshd** Step 4: Test SSH Passwordless Login from 192.168.0.12 From now onwards you can log into **192.168.0.11** as a **sheena** user from server **192.168.0.12** as a **tecmint** user without a password.  $ ssh [sheena@192.168.0.11](mailto:sheena@192.168.0.11)    -  SSH Remote Passwordless Login  Source:-  https://www.tecmint.com/ssh-passwordless-login-using-ssh-keygen-in-5-easy-steps/ |