

### Sesión 3

Vamos a tratar sobre firewalls y ssh.

#### Firewall

Permite o deniega el acceso de mensajes de determinadas ip. Se implementa en iptables y se provee un front-end. Por defecto, dejás todo abierto pero para entrar lo desactiva todo, en servicios aparece solo luego trabaja sobre tcp. Los cambios se hacen en memory y se borran al reiniciar.

2 comandos para escribir firewall-cmd

- --state : estable (running, sleep...)
- --list-all : nos da las reglas
- --add-services=(servicio) → abre el puerto determinado
- --remove-to-permanent → los cambios realizados a permanentes
- --add-port → abre un puerto determinado
- --permanent → el cambio que hagamos lo salva a disco pero no lo aplica  
↳ • --reload → se hace después para aplicar los servicios
- --get-services → nos dice los servicios

Podemos usar ~~systemctl~~ → firewall...  
Si miramos en /etc/firewall-cmd / .... podemos ver cómo actualiza servicio permanentemente

Para probar que se han aplicado las reglas hacemos uso de curl (dominio ip)

y escuchas los puertos de la ip asociada. Para aumentar las velocidades podemos usar (-T) <sup>costo</sup> <sup>seguro</sup>

o si sabemos que trabajamos en redes locales también podemos usar comando

## Ejercicio a realizar

Para instalar los serv. HTTP  $\rightarrow$  dnf-install

solo

Es una app de terminal remoto segura, una aplicación más antigua que segura es telnet

telnet <servidor\_ip>

login:  
password:  $\rightarrow$  comandos en remoto

El problema de telnet es la existencia de man-in-the-middle, pues telnet no confirma.

Como alternativa surge solo que hace lo mismo pero de forma segura



Toda la sesión es cifrada usando algoritmos de criptografía

- Clave simétrica: Quienes quieren comunicarse usan una clave secreta compartida

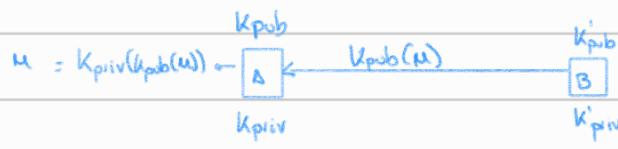
entre ellos. El problema es que es poco escalable pues hay mucha gente  
hay dos opciones

1. Todos tienen la misma llave  $\Rightarrow$  si se rompe todo se descubre

2. Claves por pares  $\Rightarrow$  poco escalable

El más común es DES o TripleDES

- Clave asimétrica (o llave pública-privada). Cifrar con una y descifrar con la otra



Si B comunica con A, B cifra con Kpub

luego no es vulnerable a man-in-the-middle  
pues no conoce Kpriv (garantiza confidencialidad)

Hemos visto la firma digital, repasar FP teoría de seguridad. También Hash

Hash  $\rightarrow$  el más usado es SHA256, el resultado del algoritmo se expresa en hexad y devuelve la misma cantidad recibida

El cuadro es un hash malo pues es predecible, y un buen algoritmo cumple:

- Es **caótico**; un cambio en el cálculo genera un resultado distinto completamente

- Es irreversible.

Se usa para el mensaje de criptomercado. Sirve para garantizar y la autoría del mensaje; para ello se encierra:

? no repudio

Form Upriv ( $M + \text{hash}(M)$ )  $\rightarrow$  El receptor calcula el hash de  $M$  y si son iguales sabe que procede de mí y el mensaje no ha cambiado

Esto es una firma digital. El más común es RSA

CERTIFICADO DIGITAL  $\rightarrow$  los proveen las autoridades de certificación

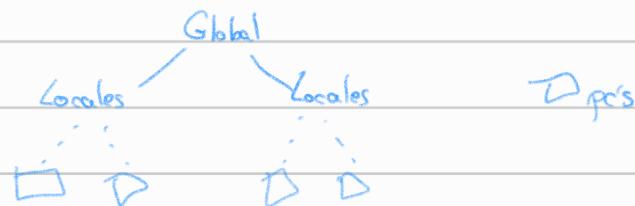
$\hookrightarrow$  El formato más común es X.509

Es un documento que, entre otras cosas, tiene:

- Mi clave pública

- Una firma de la autoridad de certificación

Para saber que la autoridad de certificación es la buena sigue una estructura jerárquica (cadena de certificación)



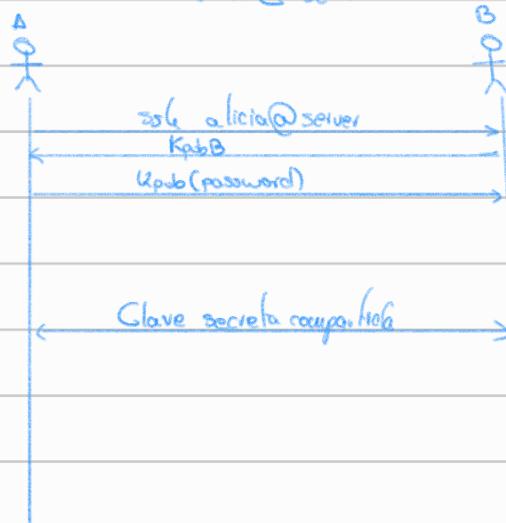
Para saber info buena www y res y pulsar F5  $\rightarrow$  me verif. y conozco el certificado.

Hasta llegar aviso conocido; son conocidos porque cuando se instala el browser lleva incorporados bastantes ya

Volvemos a la conexión ssh.

quiero conectarme al usuario alicia en server.

ssh alicia@server

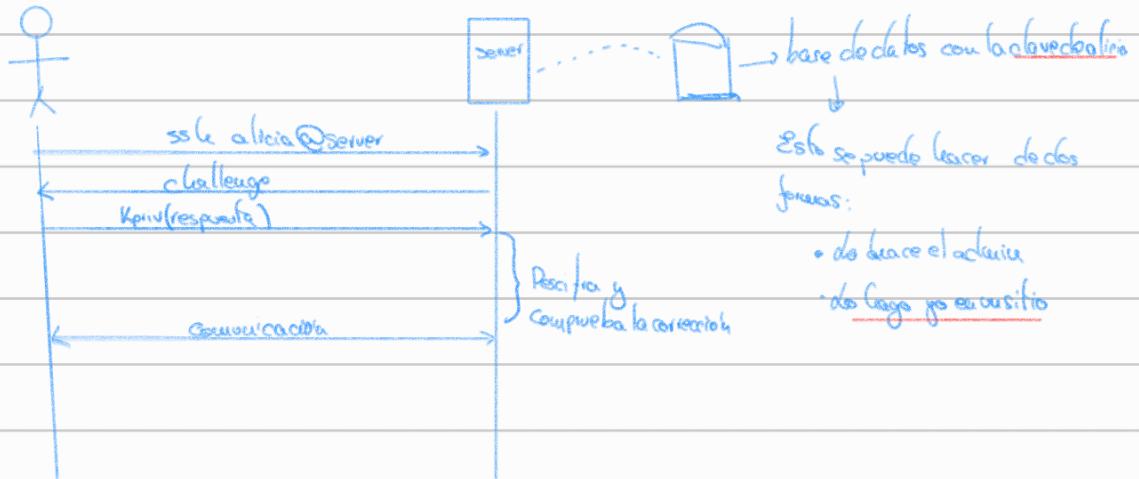


Este es un ejemplo, normalmente es más cumplido y se suele hacer un proceso de handshake y trabajar con llave simétrica.

Luego a mayor posteo se hace con llave simétrica

En linux cuando logramos solo `user@server` preguntará si acepta la conexión con la clave pública. Tras confirmaros una vez, se crea un archivo oculto que se llama .known\_hosts en el directorio .ssh:

Otra forma de conexiones con autenticación:



Para generar los claves se usa "ssh-keygen"

↳ keygen → nos dice dónde estás y cómo se llamarán

→ pide contraseña para protegerla → claves

Dobremos guardar la contraseña en el home del usuario para garantizar que ese usuario maneja.

- `ssh-add` → pide la contraseña de la clave y usó la que haserto más, modo de sesión intera

- `ssh-copy-id` → validad si esas de la identidad `user@server` → nos piden la contraseña con la contraseña de la clave.

Siguiendo autores lo guarda y hace fija.

Lo que ocurre en el server el archivo authorized\_keys contiene los usuarios que admite guardando sus llaves ⇒ podemos hacerlo simplemente copiando y pegando.

Nos permite manipular un entorno bajo una conexión segura. En esto se basa Ansible

### Ejercicio

Para cambiar el puerto sobre el que corre ssh (hay que cambiar el firewall y los puertos a usar lo suyo es forward [43000-43999])

- Configuración ssh → `/etc/ssh`

↳ `sshd_config`

↳ `sshd_config` → para el servidor (firewall)

↳ `#Port` → modif/rror y editar el comando

↳ Hay que avisar al SGIRUVR → comando de abajo

↳ retutciar esclu

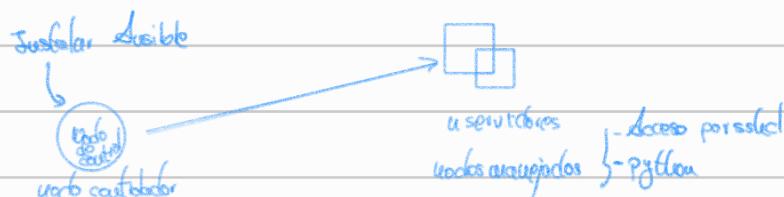
El comando para que solo -o punto usar @server

## Compositiles

scp archive user@server: path

sftp permite gestión de archivos solo

Susible (material importante)



• Recuerda el host para el sitio → en la web las páginas están en el directorio `public_html`.

Las crevillas quedan  
dobladas hacia el centro  
en paralelo

↳ Infraestrutura as Code → códigos de codificação

es donde yo pongo  
y si se de los de  
cada persona

síbolo → por defecto estaría vacío → se configura con el valor de la constante

lo c f g → tiene → tiene por defecto nulos y vacíos

lo f → para no dejar vacías las listas → restarán los elementos que no cumplen la condición

lo l e s t → devolverá el vectorario de divisible

lo es el vectorario por defecto → vacío

→ roles → scripts parametrizables  
↳ es una librería.

Busca primero en el inventario propio y luego busca en el inventario por defecto de la tienda. Trabaja como usuario normal bajo el sistema.

Arquivo de currículo ~/.curriculo/

• Conocer todos los servicios que podemos administrar, si no está enlaces us lo force (ip o DNS)  
los más interesante son los efectos.

[etiquetas] } Un servidor puede tener varias etiquetas  
ups o DNS

Para power vocublos simbólicos

### Sugar-beet

available-best = 0.8

espero con el que los reproductivos por su puesto

100

possible outcome

de forma correcta de trabajar con disciplinas es cuantitativo llaves psicológicas.

Para ejercitarse o conocerte, vas a usar visible symlinks - un módulo - a argumentos  
que te permite ver todo, se ref 43 el contenido.  
Lo buscamos visible binary  
- Si pongo all binary todos los equipos del inventario  
- Puedo poner etiquetas.

Otro comando muy usado es shell, usalo para ver si estás en modo privilegiado. Esto  
es muy desaconsejado usar los comandos de shell. Si pongo visible con shell

Si el módulo es privilegiado tendrás dos problemas:

- Tengo que hacer sudo pero hay una forma de hacerlo no recomendada
  - ↳ Nos conectamos al equipo y miramos en /etc/sudoers ; %username es su grupo y los que no son usuario → Descubrir la línea #%username !password
- Este práctica es mala así que se hace una nueva entrada nombreando pwfile por el usuario que queremos. Para que recargue el archivo usuarios visudo → cuando salvemos se aplicará directamente.

- En el ejercicio no podemos → editar fuera ese archivo y luego recargar a visudo para que lo cargue comportándose como debe.

Foto para general posible

Para posarle es sudo debemos poner --become al final y lo toma como sudo. Así, podemos usar sudo pero porque es mejor pues no todas las configuraciones de sudo usan sudo.

Cuando ejecutamos una orden contra un servidor, si tiene los mismos parámetros se quedará en el mismo estado. Es decir, los resultados idénticos; es decir, si ya se lo ha dicho, no lo hará de nuevo. Hay que intentar que los scripts sean idénticos!

Al modificar sudoers debemos comprobar si la línea está para los actualizados módulos.

Playbooks (ref 45 es más interesante) → mirar YAML syntax

Secondo scripts, es editable. (son formato.yaml) con formato anterior:

Sceny imp → ---

name: Nombre del Playbook

↳ var \$f reemplaza el valor

hosts: servidores del inventario a los que afecta

vars files: ./vars.yaml

tasks: pasos → ¡Dar nombres a todos los pasos! ir una a una

- paquete → usar todo el nombre para evitar colisiones

msg: comando

↳ Usar una opción que combine usar yaml y playbook e inventario en uno solo  
↳ las sintaxis

Para ejecutar el playbook → ansible-playbook -i inventario -c variable="...". archivo de playbook

Como el comando es complicado, debemos crear un script de Bash que contenga las líneas de ejecución

Para probar el contenido de un inventario hosts.yaml → ansible-inventory -i inventario. (-graph)

↳ ansible-facts [architectura]??

Una variable importante es ansible-facts, es mejor meterle args para que funcione mejor

↳ esto que recopila al final

→ No mezclar playbooks en un mismo archivo! Hacerlos lo más sencillos posibles

Ansible

Es un fichero externo de variables, se puede poner en el playbook (estático) o por ssh (dinámico). El archivo es .var.yaml (Puedes ser prácticas más sobre diccionarios).

Register:

↳ el valor nombre y guarda el resultado de una tarea. Es obligatorio seleccionar stdout o stderr

Root no puede acceder por ssh, para habilitar el acceso tenemos que crear un usuario, es el primer playbook que tenemos que hacer.

↳ /etc/ssh/sshd.config tenemos que cambiar el PermitRootLogin por la 'l'-password

y password.

↓ partir de a continuación es lo que tenemos que entregar, cada vez más. En un playbook todo lo que se ejecuta

de datos de todos la curiosa tiene cada uno que poseen su contenido

Los forums nos permiten descargarnos información con archivos .tar.gz

En el playbook nos creamos la llave que los users nos daban (en nuestro caso es lo que queremos)

Quitar el permitir Root login