**Práctica 3. Hashing, Cifrado y Certificados**

***Parte 1***

**1.A partir de la página de manual sobre la forma de uso de md5sum y sha1sum (y documentación extra que pueda obtener sobre MD5 y SHA-\*), elabora un resumen de cómo se emplean estas aplicaciones para generar hashcodes a partir de documentos. Discuta brevemente las debilidades conocidas y reportadas.**

Antes de comenzar con las propias preguntas del apartado voy a comentar brevemente información sobre las funciones de HASH.

* Son funciones criptográficas
* Características:
  + Son funciones de una sola vía f (texto) -> resumen/código hash, es decir, desde el punto de vista computacional (tiempo y procesador) es muy fácil obtener dado el texto claro obtener la función de hash, pero dado un resumen/código hash es muy complicado obtener el texto que ha generado ese resumen.
  + La longitud del resumen es la misma independientemente del tamaño de la entrada. Aunque la entrada sea muy grande la salida siempre van a ser 32 caracteres en hexadecimal (md5) o 40 caracteres en hexadecimal (sha1).
  + ¿Qué garantizan? La integridad de la información, es decir, que la información no se pueda alterar.

La manera de utilizar md5sum y sha1sum es la misma, es decir el comando es exactamente igual para ambas formas de creación de hashcodes, únicamente cambia el nombre de la función que utilizas. Además, existen dos formas para codificar, la primera es pasando el texto plano que vamos a querer que md5sum o sha1sum codifique mediante una tubería.

Texto

Descripción generada automáticamente

La otra posible opción es utilizando un archivo ya creado previamente, pasándole a md5sum o sha1sum el nombre de este. En este caso voy a utilizar testtttttt.pdf que ya estaba creado previamente en la máquina virtual.

Texto

Descripción generada automáticamente

En cuanto a las debilidades conocidas de estas dos formas de crear hashcodes, ambas presentan debilidades de colisión, lo cual implica que, para dos archivos claves, palabras etc. distintos, puede generar el mismo código hash. Como se ve

en esta tabla, también md5 utiliza 128 bits mientras que sha1 utiliza 160 bits, lo que influye en el número de caracteres 32 y 40 respectivamente.

|  |  |  |
| --- | --- | --- |
| HASH | Tamaño salida | Colisiones |
| MD5 | 32 caracteres hexadecimal  128 bits | SI |
| SHA1 | 40 caracteres hexadecimal  160 bits | SI |

**2.Use ambas herramientas para construir el hash asociado a diferentes tiras de caracteres que difieran en pocos caracteres. Analice y documente las diferencias entre ambas.**

Para este apartado voy a tomar las cadenas: “cadena 1” y “cadena1” que únicamente difieren en un espacio.

Primero utilizando md5 para esas dos cadenas vemos que el código hash creado es diferente puesto que las palabras a codificar eran distintas.

Texto

Descripción generada automáticamente con confianza baja

**Texto

Descripción generada automáticamente**Luego igual, pero utilizando sha1, también podemos ver que la codificación es distinta.

Y comparando ambas podemos ver, como ya hemos comentado en la tabla del apartado anterior que md5 codifica devolviendo una cadena de 32 caracteres en hexadecimal, 128 bits. Mientras que sha1 devuelve una cadena de 40 caracteres en hexadecimal, 160 bits.

**3.Elija cualquier fichero presente en tu ordenador (puedes crearlo) y obtenga el hashcode usando ambas herramientas, describiendo la salida que se obtiene.**

He creado un archivo de texto plano llamado claves.txt, cuyo contenido son 3 claves bastante comunes, sobre este fichero realizaremos ambas herramientas y después lo modificaremos levemente para ver si la salida de las herramientas sigue siendo la misma o ha cambiado.

Primero el contenido del archivo claves.txt:

**Texto

Descripción generada automáticamente**

Y ahora ejecuto md5 sobre el fichero claves.txt y después sobre claves.txt añadiendo un retorno de carro. En la flecha amarilla modifico el contenido del fichero, añadiendo el retorno de carro “\n”.

**Texto

Descripción generada automáticamente**

Y ahora exactamente lo mismo, pero con sha1, primero la codificación con el fichero original y después levemente modificado.

Texto

Descripción generada automáticamente

Como se puede ver tanto para sha1 como para md5 los códigos hash generados son distintos, eso quiere decir que el archivo ha sido modificado, cosa que hemos hecho.

La diferencia que apreciamos entre sha1 y md5 como ya hemos comentado es el número de caracteres en hexadecimal que obtenemos en la salida, 40 y 32, es decir 160 y 128 bits respectivamente.

**4. Sobre los hashes generados en el punto anterior, utiliza la herramienta hashid para tratar de obtener cuál ha sido la función de hash utilizada para crear los resúmenes anteriores.**

Como no tengo la herramienta hashid, he intentado instalar en mallet pero no he podido. Por tanto, como en el punto 6 nos pide instalar un sistema Ubuntu 20.04.2 LTS he decidido instalarlo ahora y realizar este apartado ya en este sistema.

Para instalar la herramienta en esta nueva máquina he introducido el comando *“sudo apt install hashid”.* Una vez instalada la herramienta voy a coger los códigos hash creados previamente y con el comando “*hashid código/resumen*”, va a analizar ese código hash que hemos puesto y la herramienta nos va a decir con que posibles funciones de hash ha sido creado ese código.

Por ejemplo, para el punto anterior utilizamos hashid y le ponemos como argumento el código hash generado por md5 en el apartado anterior y vemos que comienza a analizar:

Texto

Descripción generada automáticamente

Como se puede ver, ha localizado que dicho código hash, fue creado con md5, pero también pudo ser creado con otras funciones hash.

Ahora lo mismo, pero cogiendo el código hash generado por sha1.

Texto

Descripción generada automáticamente

De nuevo identifica que una de las posibles funciones creadoras de dicho código hash es sha1.

**5. A partir de la información sobre el fichero (propiedades) que se pueden obtener, por ejemplo, con la orden ls -l, genere un hashcode. Modifique alguna propiedad del fichero y vuelva a obtener el hashcode. ¿qué conclusiones obtienes?**

**Texto

Descripción generada automáticamente**

Primero realizamos un hasheo con md5 de las propiedades del fichero claves.txt y vemos que obtenemos un código hash. Después modificamos los permisos del fichero con chmod y realizamos de nuevo el hasheo y ahora vemos que el código hash generado es distinto al anterior.

Y lo mismo podemos observar con la otra función hash (sha1). La conclusión a la que llego es que modificar cualquier propiedad del fichero hace que el código hash cambie, esto puede ser de gran ayuda para ver si alguien ha modificado cualquiera de las propiedades de este con intención de manipularlo.

**6. Sobre un sistema Ubuntu20.04.2 LTS, crea dos usuarios (root/toor). (admin/123456). Determina cuál es la función de hash utilizada para que las claves de los usuarios no se almacenen en texto plano y realiza un ataque de fuerza bruta sobre las contraseñas de los usuarios almacenadas por el sistema operativo en el fichero /etc/shadow. ¿Para qué puede ser útil la herramienta John The Ripper?**

Ahora utilizo el mismo sistema que ya he montado en el apartado 4 para realizar el ejercicio con hashid.

Para crear un usuario en Ubuntu el comando es *“sudo useradd nombreusuario”* y para asignar a ese nuevo usuario una contraseña el comando es *“sudo passwd nombreusuario”.* Primero creo el usuario root (que ya está creado en el sistema) y le asigno la contraseña “toor”:

Texto

Descripción generada automáticamente

Y lo mismo para el segundo usuario “admin”, como este ya no estaba creado previamente el mensaje de aviso anterior *“useradd: el usuario <<root>> ya existe”* no lo muestra, y le asigno la contraseña “123456”:

Texto

Descripción generada automáticamente

Como hemos visto en clase, el fichero /etc/passwd contiene los usuarios del sistema y el fichero /etc/shadow las contraseñas de cada uno de los usuarios, pero a este fichero solo se puede acceder con permisos de superusuario o perteneciendo al grupo shadow.

John The Ripper es una herramienta que crackea por fuerza bruta las contraseñas, por tanto, para ver si John The Ripper consigue crackear las contraseñas de los usuarios que acabamos de crear, voy a pasarle un archivo que contenga las contraseñas hasheadas.

Para conocer cómo se almacenan las contraseñas en un sistema ejecuto el comando “*sudo cat /etc/shadow”* y elijo de ahí la de root y admin ya que son los dos usuarios que acabo de crear. Ambas líneas las introduzco en un fichero para luego pasárselo a Jonh The Ripper para que las crackee.

Creo el fichero contrasenas.txt que contiene la salida obtenida para los usuarios root y admin después de hacer *“sudo cat /etc/shadow”*

Texto

Descripción generada automáticamente

Antes de realizar el análisis con John The Ripper, vamos a ver que función de hash se utiliza para cifrar las contraseñas en el archivo /etc/shadow. En clase dijimos que $6$ se correspondía al cifrado utilizando SHA-512 pero he buscado información sobre el cifrado de contraseñas (<https://blog.elhacker.net/2022/02/icheros-etc-passwd-shadow-y-group.html>), donde aquí podemos los distintos $id que se utilizan en el cifrado de estas, y corroborar que $6$ se corresponde a SHA-512:

Texto

Descripción generada automáticamente

También utilizando la herramienta hashid utilizada en el apartado 4, donde le pasas como argumento el código hash entre comillas simples, y obtienes que la función hash que lo ha codificado es sha512. Como se puede ver en esta captura.

Texto

Descripción generada automáticamente

Después compruebo si está instalado John The Ripper y no lo está, por tanto, *“sudo apt install john”*. Ahora con el comando “john contraseña.txt” John The Ripper comenzará a crackear las contraseñas y la salida que obtenemos es:

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

Como podemos observar la herramienta ha conseguido crackear ambas contraseñas y nos dice que la contraseña para el usuario root es toor y para el usuario admin es 123456

**7. Construya un programa [buildhash] (shell script o como prefiera), que obtenga para todos y cada uno de los ficheros cuyos nombres aparecen (uno por línea) en un fichero de texto de entrada dos hashcodes: uno asociado al propio fichero y otro a sus propiedades. Para cada fichero, se generará una línea que contenga el nombre de fichero, el hashcode del fichero y el hashcode de sus propiedades, separados por ;'**

He decidido construir el código en Python, voy a dejar una captura del código tanto en este apartado como en el siguiente pero luego los adjunto en la entrega.

La justificación por la cual utilizo sha256 en vez de sha1 o md5, es porque como ya comentamos en clase realizando la tabla de la parte inferior, sha256 no tiene problema de colisiones, mientras que md5 y sha1 si, y utilizo sha256 en vez de sha512 porque utiliza la mitad de los bits en el hasheo y también menos caracteres hexadecimales.

|  |  |  |
| --- | --- | --- |
| HASH | Tamaño salida | Colisiones |
| MD5 | 32 caracteres hexadecimal  128 bits | SI |
| SHA1 | 40 caracteres hexadecimal  160 bits | SI |
| SHA256 | 64 caracteres hexadecimal  256 bits | NO |
| SHA512 | 128 caracteres hexadecimal  512 bits | NO |

Captura del código creado:

Texto

Descripción generada automáticamente

**8. Construya un programa [checkhash] (shell script o como prefiera), que tome como entrada el fichero generado por el anterior y compruebe, para cada fichero, si se han producido cambios en el mismo o en sus propiedades, generando una salida en que se muestre cada fichero y se indique si se ha modificado o no.**

Para comprobar que ambos programas funcionan voy a aportar capturas probando todas las posibilidades.

**BUILDHASH:**

****

Ejecuto el programa dos veces en la primera filesIguales contiene las rutas a dos archivos que contienen exactamente el mismo contenido, por tanto, el fichero de salida SalidaIguales, debería tener el mismo código hash tanto para el fichero como para las propiedades de este.

****

Tanto filesIguales como files3 tienen el mismo contenido.

Mientas que ahora en la segunda línea de ejecución files contiene las rutas a dos archivos que tienen distinto contenido por tanto el código hash para el fichero, así como el de las propiedades ha de ser distinto

**CHECKHASH:**

Este programa comprueba si se ha producido algún cambio en el archivo o sus propiedades, o en ambas. Para ello toma como entrada el fichero de salida del programa anterior, halla el código hash del fichero y de las propiedades de nuevo y compara con las que hay en el fichero de salida del programa anterior. Si alguna cambia el fichero ha sido modificado y si no pues no lo ha sido.

**Prueba ambos sin modificar:**

****

**Modificando el primero de ellos:**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza mediaFiles.txt original Files.txt modificado **Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media**

****

**Modificando ambos:**

Files2.txt original Files2.txt modificado

**Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**  **Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

****

***PARTE 2***

1. **Dibuja el diagrama de red lo más detallado posible.**

**Diagrama

Descripción generada automáticamente**



La puerta de enlace de Bob es 1.0.0.1/8 porque es la primera máquina que está en su mismo segmento de red, es decir, por la que tiene que pasar para comunicarse con Alice. Y la puerta de enlace para Alice es 192.168.1.3/24 ya que es la primera máquina por la que tiene que pasar que está en su mismo segmento de red para comunicarse con Bob.

1. **Configura la máquina mallet para que actúe como router. Tendrá dos interfaces de red, la eth4 con dirección 192.168.1.3/24 para eth4 en la red interna “Privada”, y la eth5 con dirección 1.0.0.1/8 en la red interna “Pública”. Deberá tener habilitado el bit de enrutamiento.**

Primero desde el propio VirtualBox modifico la configuración de red de la máquina mallet, que es la que en este caso va a actuar como router va a tener dos nuevas redes. Por tanto, en la parte de configuración de red de VirtualBox modificamos el nombre de la red del adaptador 1 a “Privada” y también de red NAT a red interna. Después habilitamos el conector 2 y hago lo mismo, modifico el nombre a “Publica”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

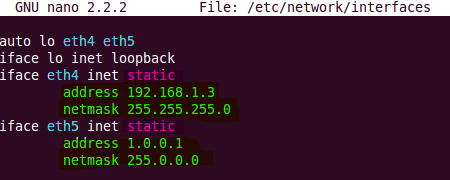
**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Ahora igual que hicimos en la práctica 1 toca modificar el fichero /etc/network/interfaces, modificando la red eth4 y añadiendo la nueva eth5 con la ip y máscara de red adecuada:

**eth4 🡪 ip: 192.168.1.3 mask: 255.255.255.0**

**eth5 🡪 ip: 1.0.0.1 mask: 255.255.255.0**

****

Ahora que ya hemos configurado las redes, tenemos que reiniciar los servicios de red con el comando *“sudo /etc/init.d/networking restart”.*

Texto

Descripción generada automáticamente con confianza media

Ahora ejecuto el comando ip a para ver si he configurado bien las interfaces de red eth4 y eth5. Y observamos en la foto que aparecen ambas interfaces de red.

Texto

Descripción generada automáticamente

Para comprobar que ambas interfaces se han actualizado y creado de forma correcta, lanzo un ping a ambas ip’s de 2 paquetes y vemos que los paquetes son transmitidos y recibidos.

**Texto

Descripción generada automáticamente**

Para habilitar el bit de enrutamiento he consultado en internet como hacerlo y en esta página web nos indica como (<https://shelllavie.com/habilitar-enrutamiento-linux/>) por tanto ejecuto el comando “sudo su” y “echo 1 > /proc/sys/net/ipv4/ip\_forward” y con este comando ya habríamos modificado el bit de enrutamiento para que todo funcione de manera correcta. Pero este comando modifica el bit de enrutamiento de forma temporal, cuando reiniciemos la máquina perderemos esta configuración.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. **Configura la máquina bob para que esté dentro de la red interna “Pública” y tenga como dirección IP la 1.0.0.2/8, y como puerta de enlace la 1.0.0.1/8.**

Ahora para que bob esté dentro de la red Pública, desde VirtualBox, y en la máquina virtual de bob, ajustes de red, y modificamos el adaptador 1, poniendo red interna, y nombre “Publica”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ahora arranco la máquina para cambiar la ip de bob:

Texto

Descripción generada automáticamente

Ahora que ya he configurado la red, tenemos que reiniciar los servicios de red con el comando *“sudo /etc/init.d/networking restart”.* Pero al hacer esto me daba error y no me modificaba la ip de bob.

Para hacerlo he asignado la ip a bob de forma temporal con el comando *“ifconfig eth0 1.0.0.2”* y para asignar la del gateway o puerta de enlace he utilizado el comando *“route add default gw 1.0.0.1 eth0”.*

Texto

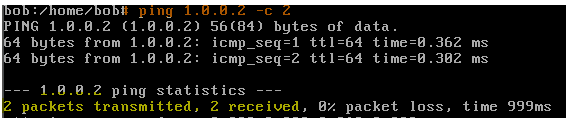
Descripción generada automáticamente

Y ahora para modificar el gateway, comprobamos con *“route -n”:*

**Texto

Descripción generada automáticamente**

Para comprobar que la interfaz se ha actualizado y creado de forma correcta, lanzo un ping a la ip de 2 paquetes y vemos que los paquetes son transmitidos y recibidos.



1. **Configura la máquina Alice para que esté dentro de la red interna “Privada” y tenga como dirección IP la 192.168.1.2, y como puerta de enlace la 192.168.1.3/24.**

Al igual que en el punto 2 y 3 modificamos primero las opciones de red de alice dentro del VirtualBox, modificando el adaptador 1 modificando el nombre a Privada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para seguir modificando la dirección ip arranco la máquina de alice e igual que en el punto 2, modifico el fichero /etc/network/interfaces y modifico su interfaz de red (eth3) con ip, máscara y gateway.

**Texto

Descripción generada automáticamente**

Y de nuevo reinicio los servicios de red con el comando *“sudo /etc/init.d/networking restart”* y compruebo con el comando ip a si la interfaz de red ha cambiado su configuración:

Texto

Descripción generada automáticamente

Para comprobar que la interfaz se ha actualizado y creado de forma correcta, lanzo un ping a la ip de 2 paquetes y vemos que los paquetes son transmitidos y recibidos.

Texto

Descripción generada automáticamente

Como en alice también le asignamos el gateway voy a comprobar si se ha modificado ejecutando el comando *“route –n“:*

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

1. **Realiza pruebas a nivel de red para comprobar que las máquinas se comunican. ¿Qué ocurre con el valor del ttl cuando un paquete atraviesa un router?**

Una vez que ya han sido modificadas todas las redes, voy a realizar pings de una máquina a otras para ver si la configuración de estas es correcta y con ello las máquinas se pueden comunicar entre sí.

Primero realizo un ping de dos paquetes desde Bob a una de las ip de mallet (1.0.0.1), después otro ping a la otra ip de mallet (192.168.1.3) y con esto hemos comprobado que bob tiene conexión con mallet, para comprobar si también la tiene con alice voy a hacer un ping a su ip (192.168.1.2) y cómo podemos ver también se completa el ping por tanto bob puede comunicarse con cualquiera de las otras dos máquinas virtuales.

Texto

Descripción generada automáticamente

alice



Ahora para comprobar la comunicación total entre máquinas, desde alice voy a realizar pings a ambas ips de mallet (1.0.0.1 y 192.168.1.3) y a bob (1.0.0.2).

Imagen que contiene Texto

Descripción generada automáticamente

bob



Como se puede observar existe comunicación total entre las máquinas puesto que los pings realizados son transmitidos y recibidos con éxito.

En cuanto al valor del ttl podemos ver que cuando el paquete atraviesa el router es decir, pasa por mallet, este se decrementa en 1 su valor, puesto que se completa un salto.

En resumen, si el ping va de bob a mallet o de alice a mallet, el valor del ttl es 64, ya que no se completa ningún salto, pero si el paquete necesita pasar por el router (mallet) es decir, ping de bob a alice o de alice a bob el ttl = 63 puesto que necesita pasar por mallet y se completa un salto.

1. **Indica las diferencias entre las reglas de tipo INPUT, OUTPUT y FORWARD en**

**iptables.**

En iptables existen distintas cadenas como INPUT, OUTPUT, FORWARD etc. Que indican por donde van a circular los paquetes dentro del sistema, y las diferencias entre ellas son:

* ***INPUT****:* hace referencia a los paquetes que tienen como destino el equipo local independientemente del origen que estos tengan. O dicho de otra forma, paquetes que entran en el host, solo entran no salen.
* ***OUTPUT***: hace referencia a los paquetes que son generados en el equipo local y que van a salir de este y de nuestra red.
* ***FORWARD*:** hace referencia a los paquetes que pasan por el equipo local, pero que son generados en equipos remotos y además tienen como destino final otro equipo/s diferente/s. FORWARD = INPUT + OUTPUT.

En forma de esquema:

Un dibujo de una cara feliz

Descripción generada automáticamente con confianza baja

1. **Configura una regla en el firewall del router para que sólo se puedan realizar peticiones de tipo ICMP (8) desde la red interna.**

Antes de configurar ninguna regla voy a ver con el comando *“sudo iptables -L”* que reglas del firewall están ya asignadas.

**Texto

Descripción generada automáticamente**

Ahora tenemos que añadir la regla que nos pide, para ello ejecutamos el comando “*iptables -A FORWARD -s 1.0.0.1/8 -d 192.168.1.0/24 -p icmp - -icmp-type 8 -j DROP*” como superusuario. El comando es “-p icmp” porque es el protocolo icmp como indica el enunciado y para decir que únicamente permita icmp 8 “- -icmp-type 8”.



Una vez creada lo comprobamos realizando el comando “*iptables -L”* y viendo si la nueva regla creada está.

Texto

Descripción generada automáticamente con confianza media

La regla que hemos creado va a impedir que un ping de bob a alice se pueda realizar ya que únicamente está permitiendo que se realicen peticiones ICMP desde la red interna, por ello voy a realizar un ping de bob a alice, y no debería de realizarse con éxito puesto que ahora el firewall va a estar denegando el aso

de estos paquetes

Texto

Descripción generada automáticamente

Como se puede ver en la captura de arriba, realizo un ping de dos paquetes de bob a alice, que son trasmitidos, pero no recibidos. Esto es debido a la nueva regla que hemos introducido en el firewall, en el router, en mallet, que deniega el paso de estos paquetes.

Ahora voy a borrar esa nueva regla creada y efectuar de nuevo el ping de bob a alice, y veremos como ahora sí que llegan los paquetes y nada los retiene. El comando indicado con la flecha es el encargado de borrar la regla.

Texto

Descripción generada automáticamente

Efectúo de nuevo el ping:

Texto

Descripción generada automáticamente

Ahora como vemos los paquetes son trasmitidos y recibidos, por tanto, la regla era la encargada de retenerlos y evitar que llegasen en este caso a alice.

1. **Configura una regla en el firewall para que sólo la máquina con IP 192.168.1.3 sea quién pueda establecer una conexión por ssh a la máquina 192.168.1.2.**

Ahora necesitamos crear una regla que permita solo a la ip 192.168.1.3 o mallet, conectarse con la ip 192.168.1.2 o alice. Para ello primero creamos las conexiones FORWARD ssh que pasan por el router. Para ello ejecuto el comando *“iptables -A FORWARD -s 0.0.0.0/0 -p tcp - - dport 22 -j DROP”* Elegimos el puerto TCP 22 porque es el que se corresponde con las conexiones ssh.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteUna vez ejecutado el comando que crea la regla, ejecutamos “iptables -L” y vemos que existen las dos reglas creadas, la del apartado anterior y esta.

Con esta nueva regla que hemos añadido al firewall nos debería de permitir realizar una conexión ssh de Mallet a Alice o de Mallet a Bob pero no de Bob a Alice y viceversa puesto que con la nueva regla añadida el router debería de denegarnos estas conexiones ssh que pasan por él.

Primero voy a hacer una conexión de Mallet a Alice que debería de permitirnos conectar, puesto que en ningún momento atraviesa el router:

Texto

Descripción generada automáticamente

Como se puede ver hemos podido establecer la conexión, ahora voy a realizar una conexión de Bob a Mallet que también nos debería permitir conectarnos, puesto que en ningún momento atraviesa el router:



Sin embargo, si ahora intentamos realizar una conexión ssh que atraviesa el router como puede ser una conexión de Bob a Alice o viceversa no debería de dejarnos puesto que el router deniega estas.

Texto

Descripción generada automáticamente



Como hemos dicho no podemos establecer la conexión ssh si esta tiene la obligación de pasar por el router, puesto que la regla FORWARD que hemos añadido las deniega.