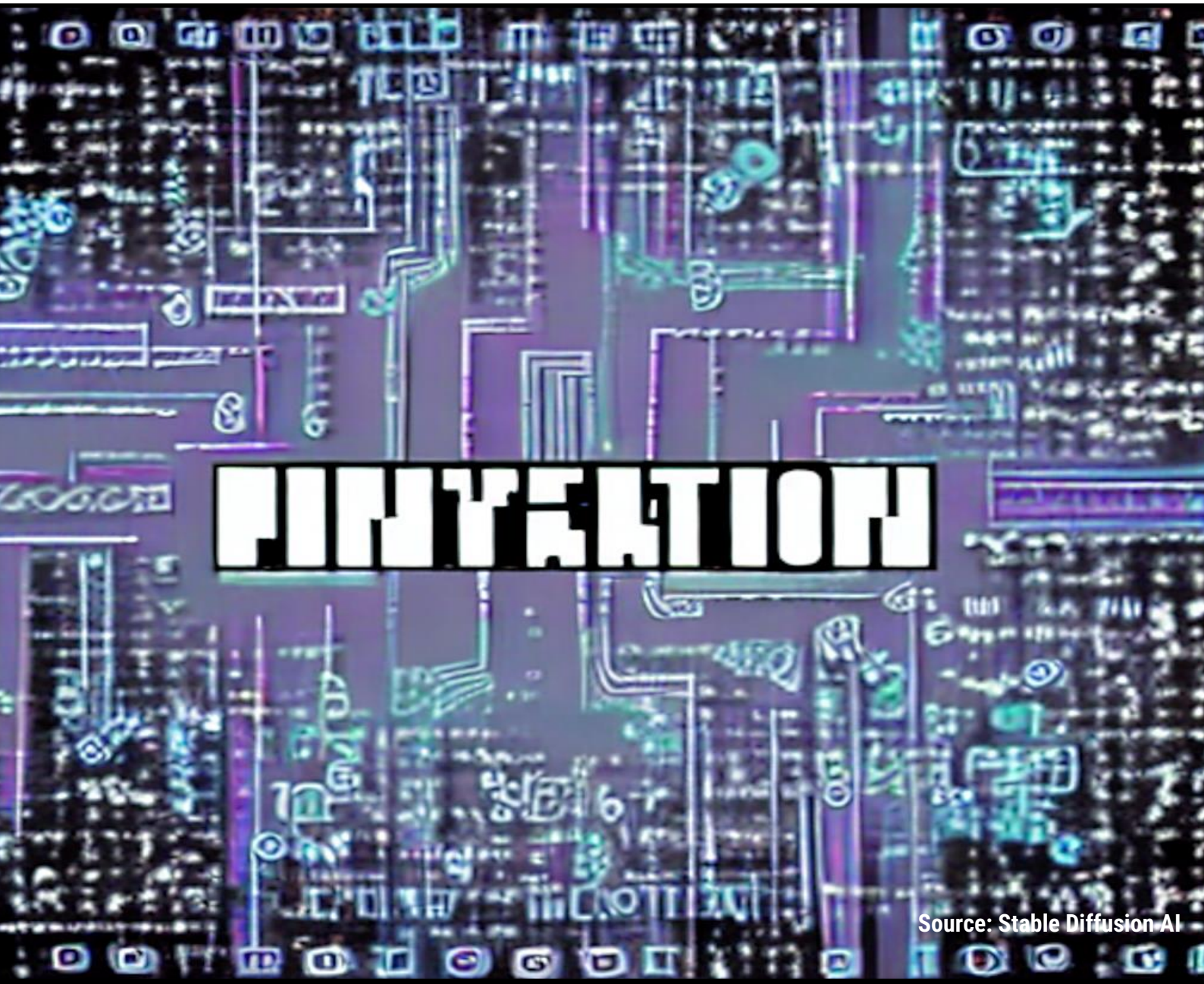


Source: Stable Diffusion AI

Laboratorio 4. Aplicaciones de la Criptografía (II)

DEPARTMENTO DE INFORMÁTICA. UNIVERSIDAD DE OVIEDO

Seguridad de Sistemas Informáticos | 2022 – 2023 (V3.1 *"S-81 Isaac Peral"*)

CONTENIDO

AVISO

Este documento forma parte de la asignatura "*Seguridad de Sistemas Informáticos*", impartida en la *Escuela de Ingeniería Informática* de la *Universidad de Oviedo*. Es fruto del trabajo continuado de elaboración, soporte, mejora, actualización y revisión del siguiente equipo de profesores desde el año 2019

- Enrique Juan de Andrés Galiana
- Fernando Cano Espinosa
- Miguel Riesco Albizu
- José Manuel Redondo López
- Luís Vinuesa Martínez

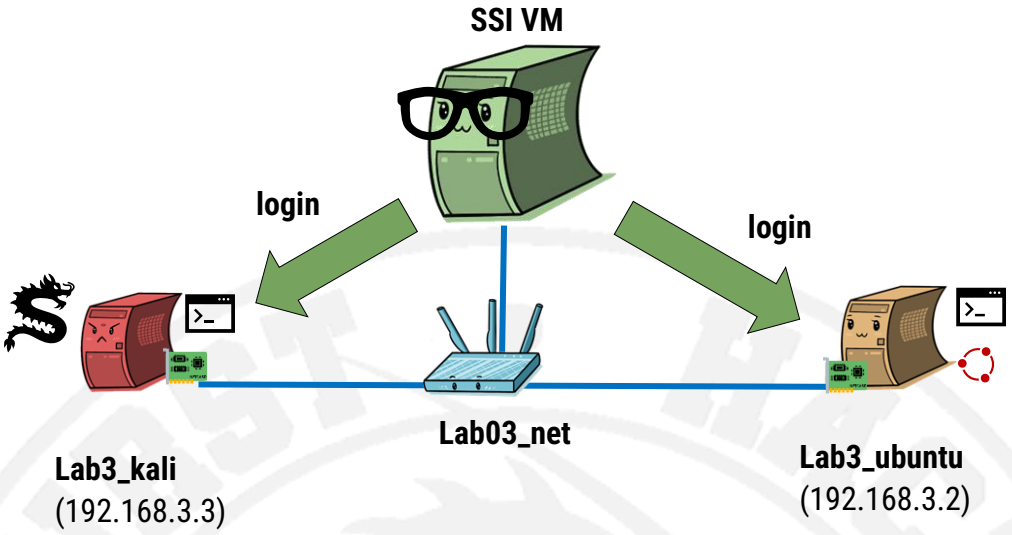
Te pedimos por favor que **NO lo compartas públicamente en Internet**. No obstante, entendemos que puedas considerar este material interesante para otras personas. Por ese motivo, hemos creado una versión de este adaptada para que pueda cursarse de forma online, disponible gratuitamente para todo el mundo y que puedes encontrar en esta dirección: https://ocw.uniovi.es/course/view.php?id=109

A diferencia de esta versión, **la versión libre puedes promocionarla todo lo que quieras**, que para eso está 😊

GRACIAS POR TU COLABORACIÓN

BLOQUE 1: GESTIÓN DE GNUPG

NOTA: Este laboratorio utiliza exactamente la misma infraestructura que el anterior. Este dibujo te recuerda cómo era:



Generar un par de claves (pública – privada) GPG

Aplicación práctica: *Puedes generar una infraestructura PKI asociada a tu cuenta de usuario de forma que ahora puedes cifrar y descifrar ficheros enviados o que vengan de personas o lugares conocidos*

En el laboratorio anterior utilizamos `gpg` (`sudo apt install gnupg`) para practicar con el cifrado de clave simétrica cifrando archivos. Ahora vamos a practicar con el cifrado asimétrico / firma digital para complementarlo. **Podemos utilizar la misma infraestructura que en el laboratorio anterior**.

Lo primero que debemos hacer es **generar un par de claves para cada usuario que participará en la comunicación.** Para generar un par de claves públicas/privadas, sigue este procedimiento:

- **GPG** necesita **entropía** para generar números verdaderamente aleatorios que se utilizarán para generar las claves. Para aumentar la aleatoriedad y acortar este proceso, te recomendamos que escribas **cosas aleatorias en el teclado** cuando la herramienta te diga que lo hagas, para que se genere suficiente entropía rápidamente y la clave se pueda crear más rápido. Otra opción para aumentar la entropía es abrir una ventana nueva de *shell* en la MV y escribir esto `sudo dd if=/dev/sda of=/dev/zero`, dejándolo en ejecución hasta que se generen las claves.
- **Te recomendamos que comiences con la máquina *Kali***, ya que la de *Ubuntu* es la única que tiene el software adecuado para aceptar claves que envíes vía `scp`. No obstante, como puedes compartir fácilmente contenido entre contenedores usando las carpetas compartidas, puedes hacer realmente lo que quieras 😊
- Ejecuta el siguiente comando para iniciar la generación de claves: `gpg --gen-key`
- Ahora se nos pedirá que proporcionemos una identidad para el titular de las claves. **Debemos tomarnos este paso muy en serio** ya que estas claves se utilizarán para identificar a cada usuario, por lo que debemos dar un nombre adecuado. Te recomendamos que uses `<tu UO>ubuntu` en el contenedor de *Ubuntu* y `<tu UO>kali` en el contenedor *Kali* en la infraestructura. Las identidades de usuario que usas con certificados **no tienen que ser las mismas que tu nombre de usuario**, ya que **son independientes**. Aunque los certificados generados se almacenen en la carpeta `/home` de cada usuario de *Ubuntu* que invoque a `gpg`, el nombre que pongas en el certificado no tiene que ser el mismo que tengas como

usuario de *Ubuntu*. Puedes decir que el nombre del certificado se usa "de cara al exterior" y el de usuario es el del SO (aunque no es raro que coincidan, siendo sinceros). El correo electrónico proporcionado puede ser falso (no conviene en un caso de uso real)

- Una vez proporcionados los parámetros, pulsamos "O" (OK) para comenzar la generación de claves
- El proceso de generación necesitará una contraseña para almacenar correctamente las claves privadas de forma segura. Pon una contraseña que puedas recordar, ya que la necesitarás varias veces más tarde. **¡No olvides que en un escenario real hay que seguir buenas prácticas de generación de contraseñas!** 😊

Resultados esperados: Esta actividad finaliza cuando se genere correctamente un par de claves para un usuario

Administrar un "llavero" (*keyring*)

Aplicación práctica: *Necesitas consultar o gestionar los pares de claves que tienes actualmente creados*

Con GPG viene el concepto de **llaveros. Cada usuario local tiene su propio llavero público y un llavero privado que almacena las claves que genera o importa de otros usuarios o sitios.** Las claves que acabamos de generar están ahora en cada respectivo llavero del usuario, que se puede gestionar mediante estos comandos:

- Para ver las claves en tu llavero privado (secreto), ejecuta: `gpg --list-keys` o `gpg --list-public-keys` para las públicas
- Si tenemos varias claves públicas pertenecientes a diferentes usuarios (como veremos en la siguiente sección), podemos ver los detalles de claves públicas concretas pasando el nombre de usuario dado al crear la clave: `gpg --list-public-keys <username>`. Lo mismo se puede hacer con las claves privadas almacenadas: `gpg --list-secret-keys<username>`

Resultados esperados: Esta actividad finaliza cuando puedes ver la clave pública y privada generada en la actividad anterior correctamente en tu llavero.

Transmitir tu clave pública a otros

Como se dijo en los temas de teoría, tu clave pública la usarán otros usuarios cuando quieran cifrar datos que solo tú puedas leer, ya que solo tu clave privada puede descifrar esos datos. **Este proceso requiere que publiques/exportes/envíes tu clave pública** para que cualquier persona interesada en enviarte información cifrada pueda acceder a ella. Por lo tanto, puedes dársela o enviarla por correo electrónico, ponerla online o almacenarla en un servidor de claves para que otros la recuperen. El procedimiento a seguir es este

- Para poder enviar tu clave pública debes exportarla primero con la opción `--export`. Para exportarla en un formato manejable utilizaremos la opción `--armor` del laboratorio anterior, y especificamos el nombre de usuario que hemos dado en la creación de claves para identificar la clave pública que queremos exportar. Ejecutamos `gpg --armor --export <nombreusuario> > <nombreusuario>_public_key.asc`, reemplazando `<nombreusuario>` por la identidad de usuario adecuada que utilizaste cuando se generaron las claves, dependiendo del contenedor con el que trabajes. **¡Cuidado porque hay un `>` en medio del comando!** 😊).

- Una vez hecho esto, se creará el archivo `<username>_public_key.asc` que contendrá tu clave pública. Te recomendamos que veas el contenido de este archivo para entenderlo mejor.
- Otros usuarios involucrados en la comunicación que quieran usar tu clave pública para cifrar datos deben importarla a su propio llavero público mediante la opción `--import` y el archivo con la clave exportada. Envía la clave al contenedor de *Ubuntu* del laboratorio e impórtala. Una vez que lo hagas en el otro contenedor, enumera las claves públicas de este segundo usuario y mira el contenido del llavero. Este ejercicio **también lo puedes hacer con un compañero enviándole tu clave pública por correo / mensajería instantánea**, incluso pegándola en el mensaje (¡para eso la exportamos en formato *armored*!)
- **NOTA:** Si se ha conectado previamente desde el contenedor *Kali* al contenedor *Ubuntu,* es muy probable que `scp` de un error que diga que el host es falso. ☹️ . Esto es normal; si se volvió a generar el laboratorio tras una ejecución anterior, los contenedores son realmente diferentes, y en la vida real esto podría significar que alguien se está haciendo pasar por el destino. Elimina los datos del sitio remoto almacenados de una ejecución anterior con `ssh-keygen -f "/<ID de usuario actual>/.ssh/known_hosts" -R "<IP del contenedor remoto>"` e intenta conectarte de nuevo. También puedes usar los directorios compartidos de los contenedores para transferir archivos.

Resultados esperados: Esta actividad finalizará cuando exportes una clave de un usuario y la importes en el llavero de otro usuario.

Validar las claves importadas mediante firmado

Cuando las claves públicas de otros usuarios se importan a un **llavero, también deben validarse** verificando su "huella digital" mediante un canal de comunicación secundario (por ejemplo, por teléfono). Cuando se verifique que la clave es correcta, debes firmarla para evitar que cada vez que la uses tengas *warnings*. Esto significa que en este modelo de funcionamiento tú eres el encargado de importar las claves públicas que recibas y también el que tiene la última palabra entre confiar en ella completamente (firmarla) o no (no firmarla). Por ejemplo, para firmar una clave pública `redondo` importada, debes seguir el siguiente procedimiento

- Editar la clave importada: `gpg --edit-key redondo`
- Esto te llevará a la interfaz de línea de comandos `gpg` desde la cual puedes mostrar la huella digital usando: `fpr`
- Estos son los datos que deben verificarse con el propietario utilizando un canal de comunicación secundario, ya que ambos deben tener la misma "huella digital" (*fingerprint*) asociada a la clave que se va a firmar. Si es así, puedes firmar la clave usando `sign`. En este laboratorio vamos a **suponer que la huella digital es correcta** y la firmamos.

Sin embargo, en esta fase **puedes encontrarte con que firmar la clave es imposible para el segundo usuario.** *¿Puedes adivinar por qué? ¿Qué se necesita para permitir a los usuarios firmar archivos utilizando criptografía de clave pública (ver tema de teoría)?* Si es tu caso, resuelve el problema y firma la clave. Para salir de **GPG** escribir `quit` o pulsar Ctrl+d. Si se pregunta si se desean guardar los cambios, debes responder `yes` para terminar de validar la clave pública. Una vez esto haya terminado, puedes editar la clave nuevamente y verificar que **GPG** informa que esta clave está ahora firmada con `signed: 1`.

ADVERTENCIA: La generación de pares de claves **REQUIERE** que el usuario que invoque el comando `gpg --gen-key` **POSEA** el terminal. Esto solo se puede lograr **iniciando sesión** en la máquina como este usuario. La forma más fácil de hacerlo es usar el script `enter` correspondiente o con `ssh` desde el *Kali* al contenedor *de Ubuntu.* ¡Ten en cuenta que ni siquiera el comando `su -` resuelve este problema! ☹️

Resultados esperados: Esta actividad finaliza cuando logres firmar una clave importada en el segundo contenedor, resolviendo los posibles problemas que puedas encontrar al hacerlo.

Copia de seguridad de la clave privada

Para guardar una copia *armored* (ASCII) o binaria de una clave privada, puedes usar la opción `--export-secret-keys`: `gpg --export-secret-keys <nombreusuario> > <fichero de la clave privada>`

Para importar la clave privada a un llavero privado en otra máquina (o restaurarla si la máquina original se reinstaló o ahora vas a usar otra para trabajar), puedes usar la opción `--import` así: `gpg --import <private_key_file>`. Ten en cuenta que la clave privada exportada sigue estando protegida por la contraseña que se utilizó cuando se creó.

Para probar esto, exporta la clave privada que creaste en el contenedor *de Ubuntu* y cópiala en la carpeta `/shared` de este contenedor. Desde el host (tu máquina virtual), vete a la carpeta compartida de este laboratorio y busca la clave privada exportada allí. **En la máquina virtual,** comprueba que no tienes ninguna clave privada creada e impórtala, comprobando luego que ahora sí tienes una clave privada importada en tu llavero de usuario local de máquina virtual. Usa el *cheatsheet* GPG del laboratorio anterior para quitar la clave importada en el llavero de la MV posteriormente. **NOTA:** Si no quieres usar tu máquina virtual, puedes hacer la misma operación entre los contenedores *Ubuntu* y *Kali*. También vale hacerlo con un compañero si quieres, pero entonces piensa bien como le vas a enviar tu clave privada para que no haya escuchas indebidas durante el envío (lógicamente tu compañero sabrá la clave y su password para poder hacer el ejercicio).

Resultados esperados: Esta actividad finaliza cuando se logra importar una clave privada exportada en otro llavero, entendiendo el propósito de hacer esto en la vida real.

Comprobación de la confianza de una clave de llavero

Incluso aunque no estemos usando un llavero público en este laboratorio, podemos usar uno para verificar cómo se descargan, administran y verifican las claves. Para ello, sigue este procedimiento en cualquiera de los contenedores del laboratorio.

- Vete a un llavero público. Por ejemplo: http://keyserver.ubuntu.com
- Busca la clave de alguien. Por ejemplo "Richard Stallman".
- Haz clic en el **ID de usuario** de cualquier resultado no revocado. Comprender lo que se muestra.
- Busca un resultado que parezca no ser falso. **Haz clic en la clave pública** y analiza lo que ves ahora.
- Haz clic en las firmas de la clave y entiende lo que estás viendo
- Para importar la clave pública con `gpg`, usa este comando: `gpg --keyserver keyserver.ubuntu.com --recv-keys <keyID>` (el que aparece en la página web tras `rsa4096/`).
- Después de hacer eso, ejecuta `gpg --list-sig <keyID>` y determina si la clave que acabas de importar puede ser realmente fiable.
- Usa el *cheatsheet* GPG de laboratorio anterior para eliminar la clave importada del llavero local (**OJO:** se podrían haber importado varias claves públicas, ¡comprueba que has eliminado todas! 😊).

Resultados esperados: esta actividad finaliza cuando puedas examinar una clave de un llavero público y la información que proporciona dicho llavero, importarla y examinar la clave que acabas de importar.

BLOQUE 2. CONFIDENCIALIDAD Y AUTENTICACIÓN

Cifrado asimétrico para confidencialidad

Aplicación práctica: *Necesitas cifrar un fichero de manera que solo una persona concreta pueda leer su contenido*

Como se explicó en los temas de teoría, para hacer cifrado asimétrico se necesita tener la clave pública de la persona para la que desee cifrar los datos. Una vez que tengas esta clave pública del receptor en tu llavero, puedes usar los siguientes comandos para el cifrado:

- Para cifrar archivos, puedes usar la opción `-e` (o `--encrypt`) junto con la opción `-r` (o `--recipient`): `gpg -e -r <nombreusuario> <ficheroacifrar>`. Por ejemplo, si alguien quiere cifrar `archivo.txt` para el usuario `redondo`: `gpg -e -r redondo archivo.txt`
- Esto producirá un archivo cifrado llamado `file.txt.gpg` que solo el usuario `redondo` puede descifrar. Si necesitas cambiar el nombre del archivo cifrado resultante, utilice la opción `-o` (o `--output`). Por ejemplo, para llamarlo `file_for_redondo.gpg`, podrías usar: `gpg -o file_for_redondo.gpg -e -r redondo file.txt`. **NOTA:** Las claves públicas no firmadas (ver sección anterior) darán una advertencia.
- **CONSEJO:** Recuerda que escribir archivos en el directorio `/shared` de los contenedores los escribe automáticamente en una carpeta conocida en el host, y viceversa 😉. Nuevamente puedes hacer este ejercicio con un compañero.

Descifrado de un texto cifrado asimétricamente

Para que un destinatario descifre los datos cifrados, debes especificar el archivo de salida utilizando `-o` y usar la opción `-d` (o `--decrypt`). Ejemplo: `gpg -o ficherodescifrado.txt -d ficherocifrado.txt.gpg`. Al destinatario se le pedirá la contraseña de su clave privada. Si es correcta, el algoritmo de descifrado podrá hacer su trabajo y los datos originales se almacenarán en `ficherodescifrado.txt`.

Resultados esperados: Esta actividad finalizará cuando puedas enviar un archivo cifrado del usuario en la máquina *Kali* al usuario en la máquina *Ubuntu* y descifrarlo correctamente gracias a las operaciones vistas de importación / exportación de claves. También debes comprender qué claves necesita para hacer eso.

SSH sin contraseña

Aplicación práctica: *Necesitas establecer conexiones seguras automáticas con una maquina remota, de forma que no se pida ninguna password cuando se haga la conexión y ésta siga siendo segura*

Una de las aplicaciones más comunes de claves públicas y privadas es **habilitar el inicio de sesión directo a través de SSH en una máquina remota que tiene nuestra clave pública registrada y asociada a una identidad de usuario**. De esta manera, cada vez que intentemos iniciar sesión en esta máquina, nuestro par de claves pública-privada se usarán para identificarnos como este usuario de forma remota, y **no se nos pedirá una contraseña**. Para hacer esto, es necesario seguir este procedimiento para generar un par de claves públicas / privadas `ssh` (no las `gpg`, **sino un nuevo par**) por lo que cada vez que el usuario en el contenedor *Kali* se conecte al contenedor *Ubuntu* con su identidad, el inicio de sesión ocurra automáticamente y no se pida contraseña (no pongas ninguna para leer en el par de claves generado): https://linuxize.com/post/how-to-set-up-ssh-keys-on-ubuntu-1804/

NOTA: No es necesario desactivar la autenticación basada en contraseña en `ssh`, (ambas formas pueden "convivir") aunque ahora podrías hacerlo, obteniendo algunas ventajas de seguridad... 😊

Resultados esperados: Esta actividad finalizará cuando puedas conectarte desde el contenedor *Kali* al contenedor *Ubuntu* utilizando tu identidad de usuario y no se te pida la contraseña para ello.

BLOQUE 3. INTEGRIDAD E IDENTIDAD

Firmas digitales para integridad

Aplicación práctica: *Necesitas asegurarte de que un fichero que has enviado o recibido no podrá corromperse o alguien lo ha alterado*

La confidencialidad de los datos es una propiedad independiente de la integridad. Ambas se pueden usar conjunta o individualmente la una de la otra. Como se dijo en teoría, la **integridad se logra a través de las firmas digitales.** Hay dos tipos de firma digital que **GPG** puede aplicar.

- Una **firma normal,** donde los datos binarios sin procesar de la firma se incluyen con los datos originales.
- Una **firma en texto plano** donde la firma se agrega como texto legible (una firma ASCII-*amored* en *base64*).

Firmas en texto plano

Para que GPG genere una firma digital en texto plano, se debe usar la opción `--clearsign`.

- Para firmar `archivo.txt` usando este método haz esto: `gpg --clearsign archivo.txt`.
- La salida es un archivo `.asc` con el contenido original más la firma digital. (***NOTA:*** *Si accidentalmente haces* `cat` *a un archivo binario (como un archivo de certificado* `gpg` *binario), el bash podría mostrar caracteres ilegibles. Puedes corregir esto escribiendo el comando* `reset`)
- En el lado receptor, el destinatario puede verificar la firma **siempre que tenga la clave pública del remitente importada** en su llavero. Esto se puede hacer así: `gpg --verify archivo.txt.asc`.

Uso de una firma digital en binario normal

Para firmar datos normalmente, utiliza la opción `-s` (o `--sign`). Usando el mismo `archivo.txt` anterior, podemos firmar con este comando: `gpg --sign archivo.txt`. Esto producirá el archivo `archivo.txt.gpg`, cuyo contenido está compuesto por nuestros datos junto con lo que parecen datos aleatorios ilegibles (**esta es la firma binaria real).** Alguien con la clave pública del remitente puede verificar la firma exactamente de la misma forma que una firma en texto plano utilizando la opción `--verify`: `gpg --verify archivo.txt.gpg`. De nuevo, puedes hacer este ejercicio con un compañero.

Resultados esperados: Esta actividad finalizará cuando puedas firmar datos en un contenedor (texto sin cifrar y binario) (por ejemplo, el *Kali*) y verificarlo en el otro (por ejemplo, *Ubuntu)*.

Usando una firma digital y datos cifrados asimétricamente

Aplicación práctica: *Necesitas enviar ficheros con contenidos secretos y asegurarte de que ese fichero no se ha modificado durante su envío*

Un remitente puede cifrar y firmar los datos para un destinatario, **combinando confidencialidad e integridad.**

- El remitente simplemente necesita combinar los comandos que usamos para el cifrado asimétrico con la opción `-s` (o `--sign`). El siguiente comando firma y cifra para el destinatario `redondo`: `gpg -o cryptandsign.enc -s -e -r redondo ficheroaprocesar.txt`

- Ten en cuenta que de nuevo el receptor necesitará la clave pública del remitente en su llavero. La opción `-d` o `--decrypt` realizará automáticamente tanto el descifrado como la verificación de firma si un archivo utiliza ambas técnicas: `gpg -o ficheroriginal.txt -d cryptandsign.enc`

El archivo ahora está descifrado. Si el destinatario no tiene la clave pública del remitente en su llavero para la verificación, el descifrado seguirá funcionando como de costumbre, pero se mostrará el siguiente mensaje: *"Can't check signature: public key not found"*.

***(NOTA:** ¡Ten cuidado con lo que comentamos antes sobre que el usuario debe tener la propiedad del terminal, ya que también puede suceder aquí!* ☹️ *)*

Resultados esperados: Esta actividad finalizará cuando puedas enviar un archivo cifrado y firmado a otro usuario y puedas ver y verificar su contenido.

Comprobación manual de la firma de un programa descargado

Aplicación práctica: *Necesitas comprobar la integridad de un fichero que has descargado, para asegurarte de que no se ha corrompido o alguien lo ha alterado maliciosamente*

Las firmas de archivos se utilizan comúnmente para verificar que los programas / documentos / etc. no hayan sido manipulados o dañados cuando se descargan. Muchos programas incluyen instrucciones sobre cómo verificar su ejecutable contra una firma que se muestra en el sitio web (o se descarga dentro de un fichero comprimido junto con el ejecutable) para garantizar que el ejecutable no esté dañado / manipulado. Para saber cómo hacerlo, sigue este procedimiento (puedes hacerlo **en la propia máquina virtual *Ubuntu):***

- Vete a la página de descarga de un programa que ofrezca firmas de verificación de sus archivos descargados. Por ejemplo: http://archive.ubuntu.com/ubuntu/dists/bionic-updates/main/installer-amd64/current/images/netboot/
- Descarga el `mini.iso` de *Ubuntu* y comprueba su firma con la herramienta `sha256sum`.
- Vete a la página donde se listan las firmas de los programas y abre el archivo apropiado con las firmas en el formato que utilizaste: http://archive.ubuntu.com/ubuntu/dists/bionic-updates/main/installer-amd64/current/images/
- Compara tu resultado con el que aparece allí.

Resultados esperados: Esta actividad finalizará cuando puedas asegurarte de que el programa descargado no se haya modificado usando la firma de su autor.

Marcas de agua

Aplicación práctica: *Necesitas marcar una imagen con algo que la identifique, para que sea más difícil para alguien decir que es suya*

Aunque **esto no es firmar un archivo,** las marcas de agua de imágenes son un procedimiento para que una imagen con derechos de autor no se pueda robar tan fácilmente por un tercero y que reclame su autoría. Por supuesto, la imagen aún podría ser manipulada, pero eliminar una marca de agua es un proceso que impediría

que algunos usuarios sin conocimientos tecnológicos las roben. También podrían usarse para crear copias físicas de documentos con marca de agua y que no puedan distribuirse sin ella, ya que eliminarlas de las copias físicas es mucho más difícil. Podemos marcar una imagen usando el paquete *Image Magick* (`sudo apt install imagemagick` y usando su herramienta `convert`). Dos ejemplos que pintan texto rojo girado (aunque puedes cambiar fácilmente los parámetros para probar otras variantes) son:

- **Texto corto, marca de agua grande**: `convert -density 150 -fill "rgba(255,0,0,0.25)" -gravity Center -pointsize 80 -draw "rotate -45 text 0,0 <text>" <original image> <watermarked image>`
- **Marca de agua de dos líneas:** `convert -density 150 -fill "rgba(255,0,0,0.50)" -pointsize 15 -draw "rotate -15 text 0,200 '<line of text 1>'" -draw "rotate -15 text -25,260 '<line of text 2>'" <original image> <watermarked image>`

Utiliza estos ejemplos para hacer marcas de agua a cualquier imagen que tengas. **Puedes hacer esto en la máquina virtual de *Ubuntu*.**

Resultados esperados: Esta actividad finalizará cuando puedas poner texto como marca de agua a cualquier imagen que descargue de Internet.

Metadatos (¡anti-firma! 😉)

Aplicación práctica: *Necesitas investigar o eliminar los metadatos de cualquier fichero*

Por último, como decíamos en el laboratorio anterior, a veces los metadatos pueden asociar un archivo (como una imagen) con nuestra identidad de usuario sin saberlo (tiene nuestro nombre, nuestra IP, datos de uno de nuestros dispositivos...). Sin embargo, hay escenarios de uso en los que **NO DESEAMOS** que aparezca esta asociación. Significa que, aunque hayamos visto la firma de documentos para asociarlos con nuestra identidad de usuario, a veces necesitamos hacer todo lo contrario y **romper cualquier asociación**. Unas veces necesitarás probar que un fichero es tuyo y otras quitar cualquier asociación con tu identidad...todo depende del propósito, dónde se publique, el contexto...

Los metadatos son posiblemente la forma más olvidada de tener datos comprometedores, por lo que terminaremos este laboratorio enseñándote las siguientes operaciones con el popular programa de manipulación de metadatos de imágenes `exiftool` (`sudo apt install exiftool`). Ten en cuenta que hay **metadatos de archivo esenciales** (información que **debe** estar en cualquier archivo de un tipo como requisito) y **metadatos adicionales, introducidos** por un programa que los crea u otra herramienta. Lógicamente, `exiftool` solo elimina metadatos adicionales.

- **Consultar los metadatos actuales de una imagen**: `exiftool <fichero de imagen>`
- **Eliminar todos los metadatos de una imagen**: `exiftool -all= <fichero de imagen>` (cuidado porque hay un espacio después del `=`)

Utiliza estos ejemplos para consultar y eliminar los metadatos de cualquier imagen que tengas o descargues. Por ejemplo: https://biosferadigital.com/sites/default/files/archivos/AK-interior.jpg

NOTA: `exiftool` es MUY potente, y puede consultar muchos ejemplos de uso aquí si tienes curiosidad:

- https://exiftool.org/examples.html
- https://exiftool.org/exiftool_pod.html

NOTA: Los metadatos **NO tienen relación con la firma criptográfica de información que vimos anteriormente.**

Resultados esperados: Esta actividad finalizará cuando puedas leer y eliminar los metadatos no esenciales de cualquier imagen.

INSIGNIAS Y AUTOEVALUACIÓN

NOTA: Tienes una versión de esta tabla de insignias en formato editable disponible en el *Campus Virtual*. Puedes usar este archivo para crear un documento en el formato que desees y tomar notas extendidas de tus actividades para crear un log de lo que has hecho. Recuerda que el material que elabores se puede llevar a los exámenes de laboratorio.

Nivel de Insignia	Desbloqueado cuando	¿Desbloqueado?
	Localices el par de claves (público y privado) generado por GPG administrando el llavero local del usuario.	
	Veas un archivo de clave pública exportado y comprendas su contenido.	
	Importes una clave de otro usuario y la puedas ver en tu llavero.	
	Puedas firmar una clave importada, entendiendo lo que se utiliza y lo que se necesita para firmar una clave.	
	Puedas crear una firma en texto plano y binaria	
	Puedas responder a esta pregunta: *Al exportar e importar una clave secreta en otro llavero. ¿La clave pública también se importa/exporta?*	
	Sabes cómo quitar una clave pública y una clave privada de un llavero.	
	Sabes cómo comprobar la firma de un programa descargado con `sha256sum`. También sabes qué algoritmos evitar para asegurarse de que la firma no es fácilmente falsificable.	
	Puedes poner una marca de agua a cualquier imagen	
	Puedes leer los metadatos de cualquier imagen. Entiendes claramente por qué esto es importante de cara a la seguridad	
	Puedes eliminar los metadatos de cualquier imagen. Entiendes claramente por qué esto es importante por razones de seguridad	
	Puedes responder a esta pregunta: *¿Qué algoritmo ha elegido GPG para generar el par de claves? ¿Qué tamaño de clave ha usado? ¿Lo consideras seguro según lo visto en la teoría del curso?*	
	Puedes responder a esta pregunta: *¿Crees que puedes enviar tu clave pública por correo electrónico a otra persona? ¿Cómo? ¿Es un problema de seguridad o no?*	
	Puedes responder a esta pregunta: *¿Qué puedes hacer al importar una clave pública de cualquier usuario?*	
	Entiendes cuál es el propósito de exportar una clave privada y volver a importarla en otra máquina. Puedes responder a esta pregunta: *¿Deberías enviar esta clave por correo electrónico? ¿Por qué?*	
	Eres capaz de iniciar sesión como usuario a través de SSH mediante una clave pública, entendiendo el proceso.	

	Eres capaz de firmar datos utilizando firmas en texto plano y binarias y comprendes las diferencias.	
	Puedes responder a esta pregunta: *Cuando examinas un llavero, ¿qué información ves?*	
	Comprendes cuántas claves necesitas para realizar un proceso de cifrado asimétrico para un usuario determinado.	
	Puedes responder a esta pregunta: *¿Qué harías si debes eliminar los posibles metadatos adicionales de cualquier imagen en una aplicación web que tengas que poner en producción?*	
	Puedes enviar mensajes cifrados a usuarios concretos de los que has importado su clave pública en tu llavero.	
	Puedes enviar mensajes cifrados y firmados a usuarios concretos de los que has importado su clave pública en tu llavero.	
	Puedes responder a esta pregunta: *¿Qué debes hacer con tus pares de claves si se vas a otra máquina, o reinstalas la actual, para evitar la pérdida de datos?*	
	Una vez que entiendes el proceso de inicio de sesión de forma remota utilizando tu clave pública, considera que el inicio de sesión con contraseña a través de SSH se puede deshabilitar totalmente para cualquier usuario. Si hacemos eso, *¿es más seguro? ¿Evitaremos ciertos ataques? ¿Qué debemos hacer si queremos que otro usuario inicie sesión en estas circunstancias?*	
	Puedes responder a esta pregunta: *¿Es el número de firmas la única razón para confiar en una determinada clave en un llavero? ¿Qué otra información se puede encontrar que hacen que la clave no sea válida?*	
	En el proceso de crear una autenticación sin contraseña a través de SSH, *¿no podemos simplemente copiar nuestra clave pública generada a cualquier usuario remoto y suplantar a cualquiera de ellos?* Si no, *¿en qué momento hicimos algo que nos impidió hacerlo?*	
	**You can't touch this**: Tu dominio de la criptografía asimétrica te permite utilizar el cifrado asimétrico para enviar datos confidenciales de forma privada a cualquier usuario que desees, asegurando también que los datos que envías han sido inalterados. Además, puedes establecer canales de comunicación confidenciales con máquinas remotas para usuarios determinados. Por último, debido a la comprensión del proceso conceptual de firma de datos, puedes utilizar eficazmente marcas de agua para identificar tus imágenes si es necesario, y determinar si una imagen puede tener información adicional que pueda identificar algo acerca de sus autores.	