Control de acceso usando ssh y técnicas de protección

Objetivos

En esta práctica de laboratorio se familiarizará con el protocolo ssh y el uso de claves públicas y privadas para el control de acceso. Opcionalmente podrán ampliar la práctica con libpam y con fail2ban.

Introducción

Usando el protocolo SSH (protocolo Secure Shell), te puedes conectar y autenticar con servicios y servidores remotos. Con las claves SSH puedes conectarte a entre otros servicios a GitHub sin necesidad de proporcionar el nombre de usuario ni el personal access token en cada visita. También puedes usar una clave SSH para firmar confirmaciones.

Al conectarse a través de SSH, se realiza la autenticación mediante un archivo de clave privada en el equipo local y todo el tráfico se encripta. Para obtener más información sobre SSH, consulta Secure Shell en Wikipedia.

Al generar una clave SSH, puedes agregar una frase de contraseña para proteger aún más la clave. Siempre que uses la clave, debes escribir la frase de contraseña. Si la clave tiene una frase de contraseña y no quieres escribir la frase de contraseña cada vez que uses la clave, puedes agregar la clave al agente SSH. El agente SSH administra las claves SSH y recuerda la frase de contraseña.

Cuando configures SSH, necesitarás generar una clave SSH privada nueva y agregarla al agente SSH. También debes agregar la clave SSH pública a tu cuenta en GitHub antes de utilizarla para autenticarte o firmar confirmaciones.

Autenticación con claves públicas y privadas: Generación de una nueva clave para SSH y adición al agente SSH

Puedes generar una nueva clave SSH en el equipo local. Después de generar la clave, puedes agregar la clave pública a un servidor o a tu cuenta en GitHub.com para habilitar la autenticación para las operaciones de Git a través de SSH.

1. Abra Terminal.

2. Pega el texto siguiente, reemplazando el correo electrónico usado en el ejemplo por tu dirección de correo electrónico de GitHub.

```
Unset
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Esto crea una llave SSH utilizando el correo electrónico proporcionado como etiqueta.

```
Unset
> Generating public/private ALGORITHM key pair.
```

3. Cuando se te pida: "Introduce un archivo en el que se pueda guardar la clave", teclea **Enter** para aceptar la ubicación de archivo predeterminada. Ten en cuenta que si ya creaste claves SSH anteriormente, ssh-keygen puede pedirte que vuelvas a escribir otra clave. En este caso, se recomienda crear una clave SSH con nombre personalizado. Para ello, escribe la ubicación de archivo predeterminada y reemplaza id_ssh_keyname por el nombre de clave personalizado.

```
Unset
> Enter a file in which to save the key
(/home/YOU/.ssh/ALGORITHM):[Press enter]
```

4. Cuando se le pida, escriba una frase de contraseña segura. Para obtener más información.

```
Unset
> Enter passphrase (empty for no passphrase): [Type a passphrase]
> Enter same passphrase again: [Type passphrase again]
```

5. Verifique que la clave se creó haciendo cat al archivo de la clave publica.

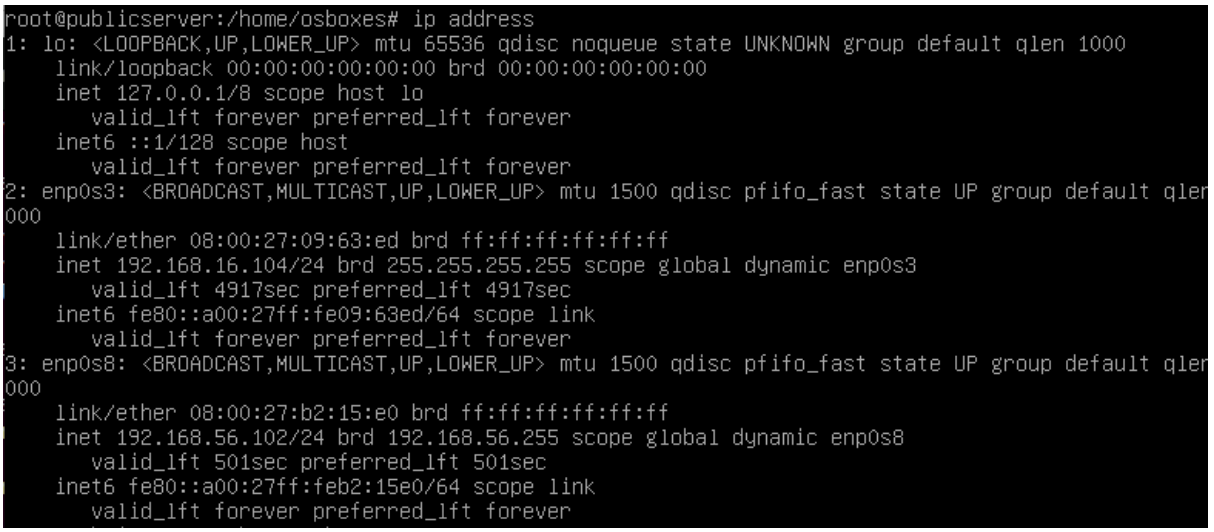
```
Unset
cat /home/YOU/.ssh/ALGORITHM

ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIIrL8HTWxcEK++pzlGrirAUJWtbbIb0qnyOmC9
7o26DS your_email@example.com
```

¿Cuáles son los algoritmos que puede utilizar ssh ? ¿RSA sigue vigente?

Subir la clave pública ssh al servidor de entrada y al segundo servidor

1. Encienda la máquina virtual “debian server”
2. En la terminal ingrese con el usuario “osboxes” password “osboxes.org”
3. verifique la ip del servidor con el comando ip address



4. Intente entrar con ssh al servidor y vuelva a salir

```
Unset
ssh root@192.168.4.71
```

¿puede ingresar con root ? ¿por qué?

5. Intente entrar con todos los usuarios del equipo

```
Unset
ssh osboxes@192.168.4.71
exit
```

6. Copiar la clave pública

Tras haber generado las claves, es hora de colocar la clave pública en el servidor virtual donde la queremos utilizar. Podemos copiar la clave pública dentro de la carpeta «autorized_keys» en el servidor virtual con la instrucción «ssh-copy-id». Para que se copie correctamente hay que indicar la dirección IP de la máquina, como se indica a continuación: Una vez finalizado este paso, accede al servidor Linux para comprobar que la configuración se haya realizado correctamente. Al acceder al servidor, si hemos generado una contraseña para la clave privada, el cliente SSH solicitará que la introduzcamos. Si no lo hemos hecho, podremos acceder al servidor sin necesidad de contraseña; únicamente con el par de claves SSH.

```
Unset
ssh-copy-id osboxes@192.168.4.71

ssh osboxes@192.168.4.71 -v

cat .ssh/authorized_keys
```

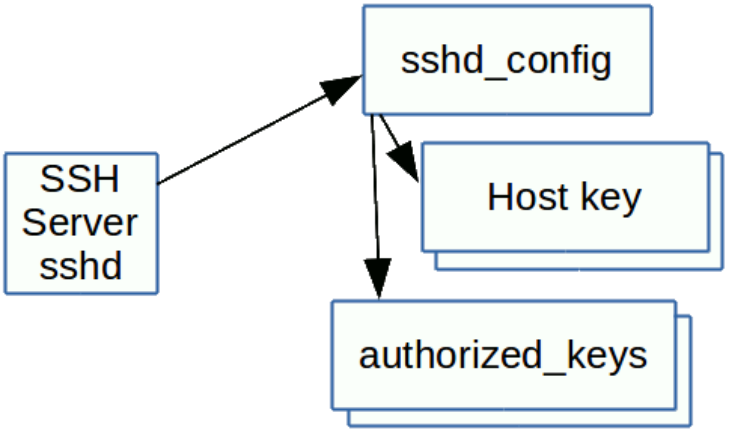
7. Impedir el ingreso con contraseña

Este último paso es opcional y sirve para mejorar aún más la seguridad. Una vez hayamos copiado las claves SSH en el servidor y nos hayamos asegurado de que podemos acceder, podemos restringir el acceso vía SSH al usuario root. Esto permite el acceso únicamente mediante las claves SSH que hemos generado. Para hacer esto tenemos que abrir el archivo de configuración de SSH:

ssh(1) obtiene datos de configuración de las siguientes fuentes en el siguiente orden:

1. opciones de línea de comandos
2. archivo de configuración del usuario (~/.ssh/config)
3. archivo de configuración de todo el sistema (/etc/ssh/ssh_config)

Para cada parámetro se utilizará el primer valor obtenido. Los archivos de configuración contienen secciones separadas por especificaciones de Host, y esa sección solo se aplica a hosts que coinciden con uno de los patrones indicados en la especificación.



Para más detalles siempre consultar “man ssh_config”

```
Unset
$ sudo nano /etc/ssh/sshd_config
```

Dentro de este archivo buscamos la línea donde aparezca «PasswordAuthentication» y la modificamos para asegurarnos de que solo se pueda acceder usando las claves SSH:

PasswordAuthentication no

Nota: Si la línea de «PasswordAuthentication» no existe, tendremos que crearla.

8. Reiniciar el servicio ssh

Por último, guardamos los cambios y recargamos SSH para que se efectúen:

```
Unset
$ sudo systemctl reload sshd
o
$ sudo service ssh restart
```

Una vez completados todos los pasos, dispondremos de una máquina virtual más segura y solo podremos acceder a ella si disponemos de las claves SSH generadas. Asimismo, para más seguridad, antes de cerrar la sesión SSH, deberíamos probar la conexión desde otro terminal para verificar que funciona correctamente.

Cree un nuevo usuario

```
Unset
$sudo adduser nonpriv

Adding user `nonpriv' ...
Adding new group `nonpriv' (1001) ...
Adding new user `nonpriv' (1001) with group `nonpriv' ...
Creating home directory `/home/nonpriv' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for nonpriv
Enter the new value, or press ENTER for the default
       Full Name []: non priv user
       Room Number []: 1234
       Work Phone []: 1234
       Home Phone []: 1234
       Other []:
Is the information correct? [Y/n] y
```

intente cambiarse a este nuevo usuario

```
Unset
 su nonpriv
```

desde la computadora real o la otra máquina virtual intente acceder por ssh con el nuevo usuario

```
Unset
ssh nonpriv@...
```

¿Puede ingresar ? ¿Por qué ? ¿Cómo podemos hacer para poder ingresar ?

AuthorizedKeysFile

Históricamente, la mayoría de las organizaciones no han tocado la ubicación de los archivos de claves autorizadas. Esto significa que están en el directorio de inicio de cada usuario y cada usuario puede configurar credenciales permanentes adicionales para él y sus amigos. También pueden agregar credenciales permanentes adicionales para cualquier cuenta de servicio o cuenta raíz en la que puedan iniciar sesión. Esto ha provocado enormes problemas en las grandes organizaciones en torno a la gestión de claves SSH.

Recomendamos que las organizaciones establezcan una gestión adecuada del ciclo de vida de las credenciales basadas en claves y establezcan las opciones relacionadas como parte de este proceso.

```
Unset
AuthorizedKeysFile /etc/ssh/authorized-keys/%u
```

Las empresas también deben prestar atención a las opciones AuthorizedKeysCommand y AuthorizedKeysCommandUser. Normalmente se utilizan cuando SELinux está habilitado y para recuperar claves SSH de directorios LDAP u otras fuentes de datos. Su uso puede hacer que la auditoría de claves SSH sea engorrosa y pueden usarse para ocultar claves de puerta trasera de una observación casual.

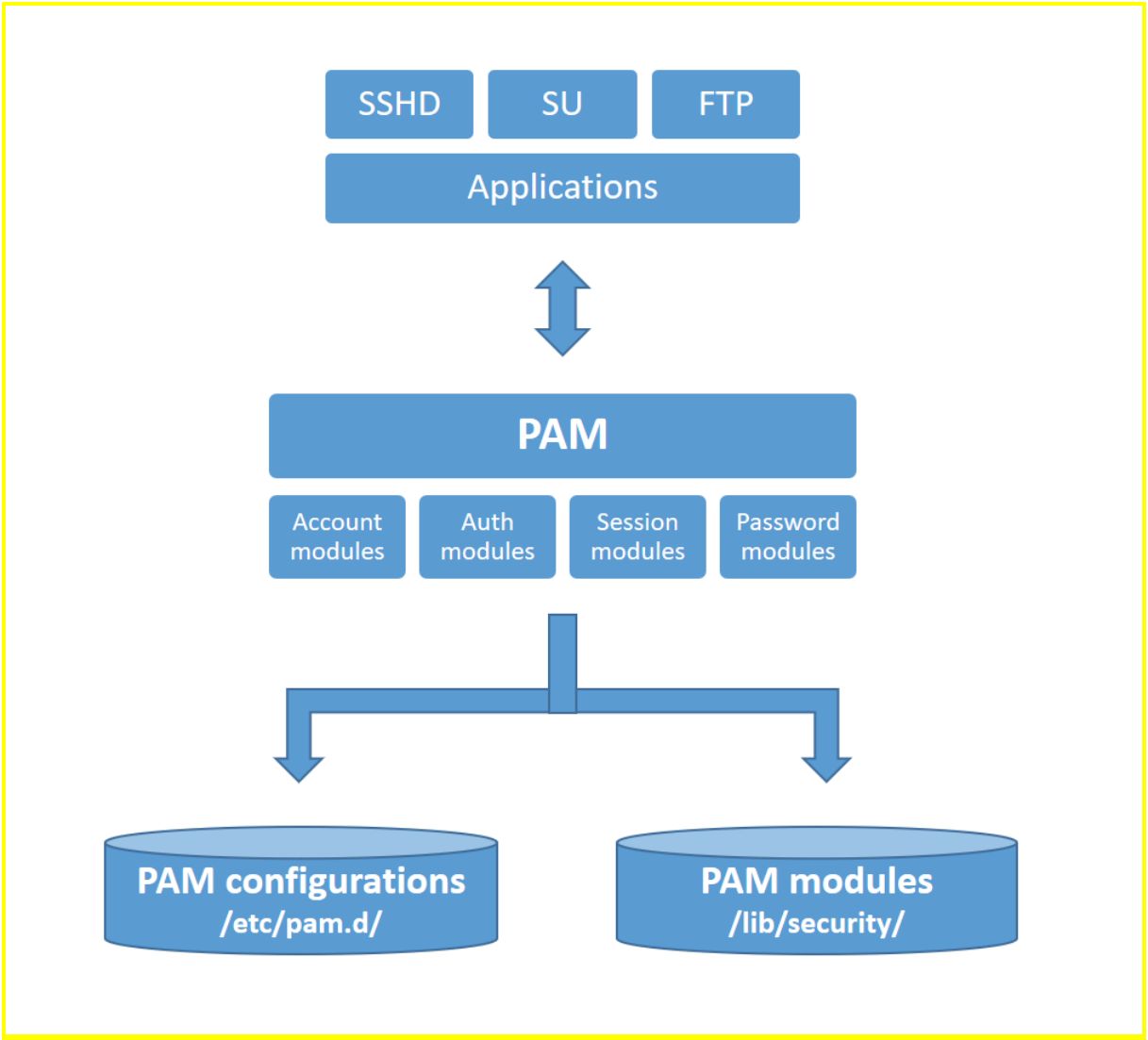
Banner

Banner de inicio de sesión Es posible que muchas empresas, especialmente en el gobierno, quieran imprimir un banner de inicio de sesión con advertencias legales antes de solicitar una contraseña. La opción Banner hace esto. Si se especifica esta opción, el contenido del archivo se imprimirá al cliente antes de iniciar sesión.

```
Unset
Banner /etc/banner
```

Como no todo tiene que ser monotono ¿Se animan a ser creativos y poner un buen ascii art?

Autorización usando PAM



Existen muchas formas de denegar el acceso de un usuario a un sistema GNU/Linux, pero quizás el empleo del módulo *pam_listfile.so* en SSH sea uno de los más efectivos y potentes.

Se puede dar el caso de que un usuario pueda tener que acceder al equipo localmente, pero de ninguna forma nos interese que éste pueda acceder remotamente. En este caso, tener **un listado de usuario al que denegar el acceso** sería más que interesante. Pues bien, esto es lo que vamos a configurar a continuación. Para ello, vamos a modificar el módulo SSH de PAM (*/etc/pam.d/ssh*) para indicarle que emplee la opción de una lista de usuarios denegados, tal como sigue.

...
auth required pam_listfile.so item=user sense=deny file=/etc/ssh/ssh.deny onerr=succeed
...

Una explicación sencilla de la línea que acabamos de agregar es:

- **auth required pam_listfile.so**
  Nombre del módulo requerido para validar los usuarios.

- **item=user**
  Chequea el campo usuario.
- **sense=deny**
  Deniega el acceso si el usuario está en el fichero especificado.
- **file=*/etc/ssh/ssh.deny***
  Fichero con los nombres de usuario (uno por línea)
- **onerr=succeed**
  Devuelve un PAM_SUCCESS si hay un error.

Una vez tenemos agregada la línea anterior, simplemente creamos el fichero */etc/ssh/ssh.deny* y escribimos los **nombres de usuario a los que queramos impedir el acceso**, uno por línea.
usuario_1
usuario_2
nonpriv
...
Desde este momento cualquier usuario que desee acceder al sistema y esté dentro del fichero *ssh.deny* no lo podrá hacer.

vuelva a intentar acceder al sistema usando ssh

```
Unset
ssh nonpriv@...
```

También es posible indicar solamente aquellos usuarios que nos interesa que puedan acceder. En ese caso la línea es un poco diferente.
...
auth required pam_listfile.so item=user sense=allow file=/etc/ssh/ssh.allow onerr=fail
...
Como se puede observar, ahora el tipo *sense* es *allow*, es decir, **los usuarios que coincidan podrán acceder**. Y para ser consistentes, usamos un nombre de fichero */etc/ssh/ssh.allow*.

Fail2ban

Si bien es técnicamente imposible tener éxito con un ataque de fuerza bruta cuando la autenticación es realizada con claves públicas y privadas, existe la posibilidad de realizar ataques de denegación de servicio utilizando fuerza bruta. Fail2ban puede mitigar significativamente los ataques de fuerza bruta mediante la creación de reglas que alteran automáticamente la configuración de su firewall para prohibir IP específicas después de una

cierta cantidad de intentos fallidos de inicio de sesión. Esto permitirá que su servidor se proteja contra estos intentos de acceso sin su intervención.

```
Unset
sudo apt update
sudo apt install fail2ban iptables
```

```
Unset
systemctl status fail2ban.service
```

```
Unset
cd /etc/fail2ban
```

```
Unset
sudo cp jail.conf jail.local
sudo nano jail.local
```

Mientras se desplaza por el archivo, este tutorial revisará algunas opciones que quizás desee actualizar. La configuración ubicada en la sección [DEFAULT] cerca de la parte superior del archivo se aplicará a todos los servicios admitidos por Fail2ban. En otras partes del archivo, hay encabezados para [sshd] y para otros servicios, que contienen configuraciones específicas del servicio que se aplicarán además de los valores predeterminados.
/etc/fail2ban/jail.local

[default]
. . .
bantime = 10m
. . .

El parámetro bantime establece el período de tiempo que un cliente será baneado cuando no se haya autenticado correctamente. Esto se mide en segundos. De forma predeterminada, esto está configurado en 10 minutos.
/etc/fail2ban/jail.local

[default]
. . .

findtime = 10m
maxretry = 5
. . .

Los siguientes dos parámetros son findtime y maxretry. Estos trabajan juntos para establecer las condiciones bajo las cuales se determina que un cliente es un usuario ilegítimo que debe ser prohibido.

La variable maxretry establece la cantidad de intentos que un cliente tiene para autenticarse dentro de un período de tiempo definido por findtime, antes de ser prohibido. Con la configuración predeterminada, el servicio fail2ban prohibirá a un cliente que intente iniciar sesión sin éxito 5 veces en un período de 10 minutos.

La siguiente es la parte del archivo de configuración que se ocupa de los servicios individuales. Estos se especifican mediante encabezados de sección, como [sshd].

Cada una de estas secciones debe habilitarse individualmente agregando una línea
enable = true
debajo del encabezado, con sus otras configuraciones.

[sshd]
. . .
enable = true
. . .

```
Unset
sudo systemctl restart fail2ban
```

intente ingresar con la contraseña equivocada múltiples veces ¿que sucede ? cual es la salida del comando “sudo iptables -L” ?

sudo fail2ban-client set sshd unbanip xxx.xxx.xxx.xxx

Reflexión

¿En qué casos conviene listar los usuarios a denegar el acceso? ¿sería más interesante listar solamente a quienes deseo permitir el acceso ? ¿En qué casos puede tener sentido ? ¿Qué otros servicios puede vigilar fail2ban? ¿Se puede configurar para que envíe alertas cuando se active?