



ELIMINA LAS PELIGROSAS «LLAVES MAESTRAS» CREANDO UN SISTEMA DE GESTIÓN DE CLAVES SSH



www.splashtop.com/foxpass



¿QUÉ SON LAS LLAVES MAESTRAS?

En muchas organizaciones, los ingenieros tienen acceso ilimitado a todos los servidores y a todos los datos que contienen. Esto se suele denominar acceso de nivel «administrador» o «root». Un acceso tan irrestricto se considera una «llave maestra», igual que la «llave maestra» de un edificio de oficinas abre todas las puertas.

Y así como perder la «llave maestra» de un edificio de oficinas puede tener consecuencias desastrosas, dar a los ingenieros «llaves maestras» de servidores y datos puede ser aún más catastrófico cuando una se pierde, es robada o se usa indebidamente.

¿POR QUÉ LAS LLAVES MAESTRAS SON UN ERROR?

CASO 1: INSIDER MALICIOSO – IMPROBABLE PERO LETAL

Un insider malicioso es un empleado que dispone tanto de los medios necesarios para acceder a los activos de una empresa como del motivo para causar daño. Este caso es muy improbable, ya que solo se concede acceso root a los empleados que se han ganado cierto grado de confianza dentro de la organización, pero este escenario no es imposible.

Pensemos, por ejemplo, en el insider malicioso que robó gigabytes de datos de Tesla y los transfirió a terceros. Según un estudio, más del 49 % de las empresas están preocupadas por los atacantes maliciosos, ya que se trata de una amenaza real y creciente.

CASO 2: INSIDER NEGLIGENTE – PROBABLE Y SIN RASTRO

A pesar de toda la formación que reciben los profesionales de TI, la mayoría de las empresas siguen siendo vulnerables y es probable que sean víctimas de phishing y de otras formas de ataques de ingeniería social. De hecho, según un estudio, más del 34 % de todas las brechas de seguridad de 2018 fueron causadas por personal interno. Además, este tipo de ataques no suele dejar rastro, y la infraestructura de una empresa puede haber sido vulnerada antes de que nadie sepa lo que ha ocurrido.

CASO 3: CREDENCIALES ROBADAS – PROBABLE Y RASTREABLE

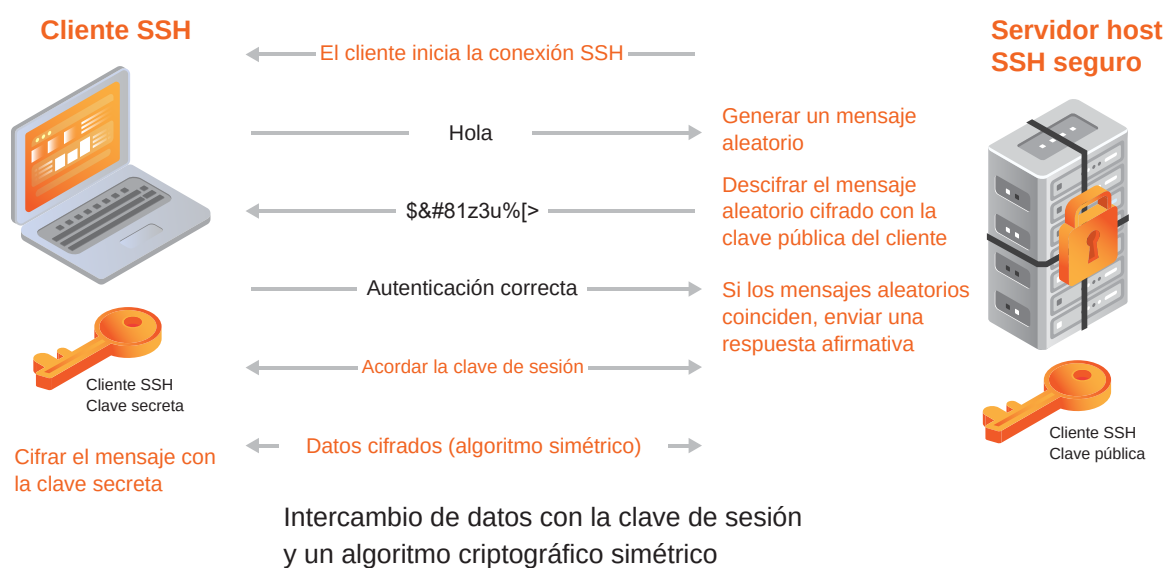
El robo de equipos portátiles o de credenciales también es un ataque externo probable dirigido a los empleados con el mayor nivel de derechos de acceso, es decir, los ingenieros y el equipo interno. Una vez que un atacante externo obtiene estas credenciales, puede acceder fácilmente a la infraestructura de la empresa. Si esas credenciales se compartían entre varios empleados, deben cambiarse de inmediato, lo que impide que todos los demás empleados accedan a los servidores hasta que se les pueda informar de forma segura de las nuevas credenciales. Si cada persona tuviera sus propias credenciales para el servidor, solo habría que restablecer las de la persona afectada.

¿CÓMO REDUCIR EL RIESGO DE FILTRACIONES DE DATOS?

Las empresas tecnológicamente avanzadas conocen los peligros de los ataques internos y restringen el acceso de sus ingenieros según sus funciones. El control de acceso basado en roles (RBAC) limita el acceso de los usuarios en función de su rol dentro de la organización.

Implementar RBAC significa que un usuario solo tiene acceso a lo que necesita con frecuencia para hacer su trabajo. Sin embargo, «con frecuencia» no significa necesariamente a diario, cada semana o ni siquiera cada mes. La opción más segura es dar a los ingenieros el nivel de acceso más bajo posible. Cuando se requiera un acceso de nivel superior, este debe supervisarse, controlarse y limitarse en el tiempo.

¿Cómo lograr un mecanismo de control de acceso bien supervisado, controlado y limitado en el tiempo? La mejor solución es un sistema en el que cada ingeniero tenga su propia clave y que permita acceder a un servidor durante un periodo predefinido: cuanto más corto, mejor.



¿CUÁNDO DEBEN CADUCAR LOS PERMISOS?

Normalmente, a un ingeniero se le conceden permisos desde que empieza en una empresa hasta que se marcha. Sin embargo, una opción más dinámica sería conceder permisos:

- Solo durante un turno de guardia
- Hasta que se cierre un ticket de tipo Jira
- Solo si el permiso se ha utilizado al menos una vez en el último mes

¿QUÉ ES UNA CLAVE SSH?

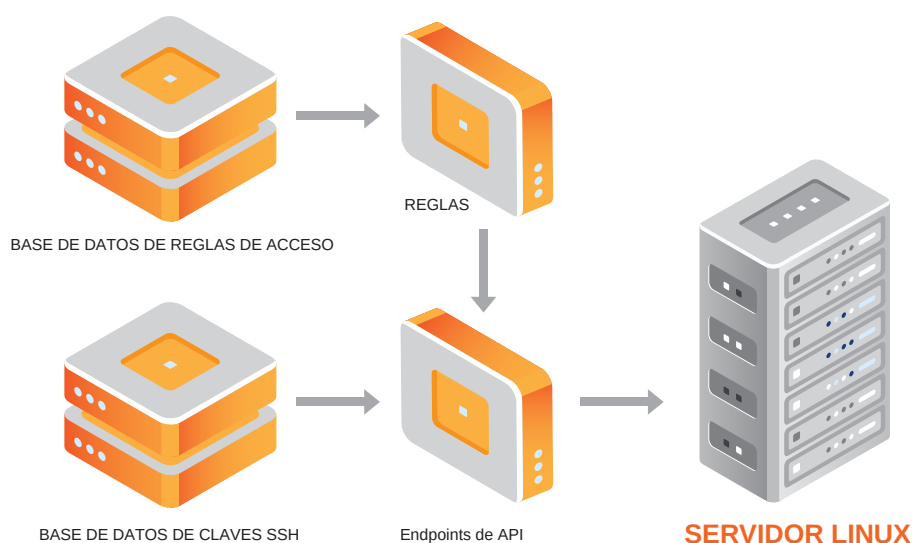
Una «clave SSH» es un mecanismo criptográficamente seguro para demostrar la propia identidad ante un servidor al iniciar sesión a través de la red. En ese sentido se parece a una contraseña larga, pero ahí terminan las similitudes. Una «clave SSH» se compone en realidad de dos partes: una clave privada en un archivo y una clave pública en otro. El contenido de estos dos archivos es muy distinto, pero se complementan entre sí y, por eso, se crean al mismo tiempo.

La mitad privada de la clave SSH debe protegerse y no salir nunca, por ejemplo, del sistema en el que se generó. La mitad pública, en cambio, puede compartirse libremente. La clave privada no puede regenerarse a partir de la clave pública. Al copiar la parte pública de la clave en cualquier servidor remoto en el que este usuario deba poder iniciar sesión, se concede acceso al propietario de la mitad privada.

¿CÓMO CREAR UN SISTEMA DE GESTIÓN DE CLAVES SSH?

Los componentes necesarios para crear un sistema de gestión de claves SSH son:

- 1 Una base de datos que almacena las partes públicas de las claves SSH de los usuarios
- 2 Una base de datos de reglas de acceso que indica qué usuarios acceden a qué servidores
- 3 Un servidor de aplicaciones compatible con aplicaciones web sencillas y endpoints de API



CÓMO CONFIGURARLO:

- 1 Tradicionalmente, el daemon sshd busca en el disco el archivo `authorized_keys` del usuario en su directorio `~/.ssh`.

Sin embargo, esta configuración aprovecha la directiva de sshd `AuthorizedKeysCommand` (disponible desde la versión 6.2 de OpenSSH) para indicar a sshd que use los resultados del script ubicado en `/usr/local/bin/get_keys.sh`. Para ello, cambia la ubicación en el archivo `sshd_config`. Ten en cuenta que, si esto falla, sshd sigue recurriendo al archivo `authorized_keys` del disco. Para hacerlo, edita `sshd_config` así:

```
ubuntu@host:~$ vi /etc/ssh/sshd_config

[. . .]
AuthorizedKeysCommand /path/to/get_keys.sh
AuthorizedKeysCommandUser nobody
```

- 2 El script `/usr/local/bin/get_keys.sh` puede escribirse en cualquier lenguaje, aunque se recomienda un script de Shell o Python. El primer y único argumento de este script es el nombre de usuario de Linux del usuario que solicita las claves. El script debe reenviar la misma consulta al endpoint de API creado. Para ello, usa el comando `curl`, p. ej. ``curl https://<endpoint>/get_keys/?user=$1``. A continuación se muestra un ejemplo de `get_keys.sh` (específico para AWS):

```
#!/bin/sh

user="$1"
hostname=`hostname`
instance_id=`curl -s -q -f
http://169.254.169.254/latest/meta-data/instance-id`
curl -s -q -f -m 5 "https://keyserver.example.com/sshkeys/?
user=${user}&hostname=${hostname}&instance_id=${instance_id}" 2>/dev/null

exit $?
```

Ejemplo de solicitud:

`https://keyserver.example.com/sshkeys/?user=aren&hostname=prod-db-1&instance_id=i-0b263919b6498b123`

3 Crea el endpoint /sshkeys/ en la API de forma que:

- obtenga el argumento del nombre de usuario de la cadena de consulta;
- busque las claves SSH de ese usuario en la base de datos y
- devuelva las claves SSH en el mismo formato que un archivo `authorized_keys`: cada clave pública SSH en su propia línea.

Ejemplo:

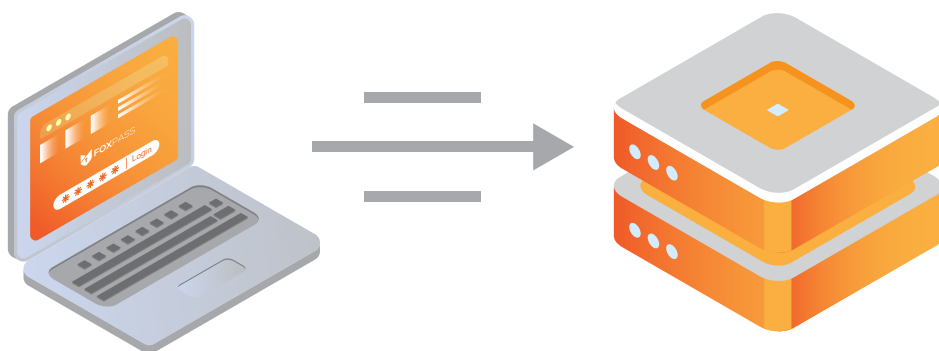
```
def sshkeys(request):
    hostname = request.GET['hostname']
    aws_instance_id = request.GET['aws_instance_id']
    username = request.GET['user']

    user = get_user(username)

    keys = []
    if is_user_allowed_on_host(user, hostname, aws_instance_id):
        keys = user.keys

    return HttpResponse("\n".join(key_text) for key in keys),
    content_type='text/plain')
```

4 Crea una aplicación web que los usuarios puedan usar para gestionar sus claves SSH. Es fácil autenticar a un usuario en la aplicación web mediante OAuth con GSuite u Office 365. El usuario debe poder añadir una clave nueva y desactivar una antigua.



Interfaz web sencilla para iniciar sesión (OAuth)
y subir las claves SSH (mitad pública)

BASE DE DATOS DE CLAVES SSH

Esta configuración básica puede ampliarse incorporando una base de datos de reglas de acceso que regulen y controlen el acceso a los servidores. Mediante la integración con API externas, el acceso puede modificarse en función de determinadas condiciones y reducirse o revocarse en consecuencia.

- Para garantizar que solo se distribuyan las claves SSH de los empleados actuales: contrastar con la API de GSuite, Office365, OneLogin u Okta
- Para imponer la rotación de claves SSH: excluir las claves con más de 90/180/360 días
- Para restringir a qué servidores puede acceder un usuario: comprobar los derechos de acceso definidos para cada usuario. Si no existe el permiso para ese servidor concreto, impedir que el usuario acceda al sistema

Ejemplo:

<https://keyserver.example.com/sshkeys/?user=aren&hostname=prod-db-1>

```
hostname=prod-db-1
«Máquina de producción» match regex = ^prod-.*
¿Tiene «aren» derechos ahora sobre alguna «Máquina de producción»?

o

hostname=prod-db-1
«Máquina de BD» match regex = -db-\d+$
¿Tiene «aren» derechos ahora sobre alguna «Máquina de BD»?

```

CONCLUSIÓN

Las posibilidades de esta configuración para la gestión de claves SSH y el control de acceso son infinitas. Las vulnerabilidades de seguridad y los ataques internos siempre son una gran preocupación para las organizaciones con datos confidenciales que pueden compartirse con ingenieros, ya que su filtración puede tener un impacto enorme. Siempre conviene dar a los empleados los mínimos privilegios y limitar el acceso a recursos críticos cuando se concede. Si Ashley Madison, Target o Tesla hubieran tenido un mejor control de acceso, es posible que no hubieran sufrido brechas de seguridad tan costosas y públicas. La gestión de claves SSH es un mecanismo útil para proteger la infraestructura y ayudar a evitar que los equipos internos filtren información privilegiada al exterior.

CÓMO FOXPASS PUEDE AYUDARTE

Foxpass ofrece productos completos y fáciles de usar para la gestión de claves SSH y el control de acceso privilegiado, que permiten a los ingenieros acceder a los servidores necesarios durante un periodo de tiempo predefinido. Además, Foxpass es un sistema completo de gestión de usuarios y grupos para Linux que se sincroniza con directorios populares como GSuite, Office365 y Okta.