



**ELIMINE AS PERIGOSAS “CHAVES MESTRAS”
CRIANDO UM SISTEMA DE GERENCIAMENTO DE CHAVES SSH**



www.splashtop.com/foxpass



O QUE SÃO CHAVES MESTRAS?

Em muitas organizações, os engenheiros recebem acesso irrestrito a todos os servidores e a todos os dados que eles contêm. Isso costuma ser chamado de acesso de nível “administrador” ou “root”. Esse acesso ilimitado é considerado uma “chave mestra”, assim como a “chave mestra” de um prédio comercial abre todas as portas.

E, assim como perder a “chave mestra” de um prédio comercial pode ter consequências desastrosas, dar aos engenheiros “chaves mestras” para servidores e dados pode ser ainda mais catastrófico quando uma delas é perdida, roubada ou usada indevidamente.

POR QUE AS CHAVES MESTRAS SÃO UM ERRO?

CASO 1: INSIDER MAL-INTENCIONADO – IMPROVÁVEL, MAS LETAL

Um insider mal-intencionado é um funcionário que tem tanto os meios necessários para acessar os ativos de uma empresa quanto o motivo para causar danos. Esse caso é altamente improvável, já que apenas funcionários que conquistaram certo grau de confiança dentro da organização recebem acesso root, mas esse cenário não é impossível.

Veja, por exemplo, o caso do insider mal-intencionado que roubou gigabytes de dados da Tesla e os transferiu a terceiros. Segundo um estudo, mais de 49% das empresas estão preocupadas com invasores mal-intencionados, pois essa é uma ameaça real e crescente.

CASO 2: INSIDER NEGLIGENTE – PROVÁVEL E NÃO RASTREÁVEL

Apesar de todo o treinamento oferecido aos profissionais de TI, a maioria das empresas ainda é vulnerável e provavelmente será vítima de phishing e de outras formas de ataques de engenharia social. Na verdade, segundo um estudo, mais de 34% de todas as violações em 2018 foram causadas por insiders. Além disso, esses tipos de ataque geralmente não são rastreáveis, e a infraestrutura de uma empresa pode já ter sido invadida antes que alguém perceba o que aconteceu.

CASO 3: CREDENCIAIS ROUBADAS – PROVÁVEL E RASTREÁVEL

O roubo de notebooks ou de credenciais também é um provável ataque externo, direcionado aos funcionários com o nível mais alto de direitos de acesso, ou seja, os engenheiros e a equipe interna. Depois que um invasor externo obtém essas credenciais, ele pode acessar facilmente a infraestrutura da empresa. Se essas credenciais eram compartilhadas entre vários funcionários, elas precisam ser alteradas imediatamente, o que impede todos os outros funcionários de acessar os servidores até que possam ser informados das novas credenciais com segurança. Se cada pessoa tivesse suas próprias credenciais para o servidor, somente as da pessoa afetada precisariam ser redefinidas.

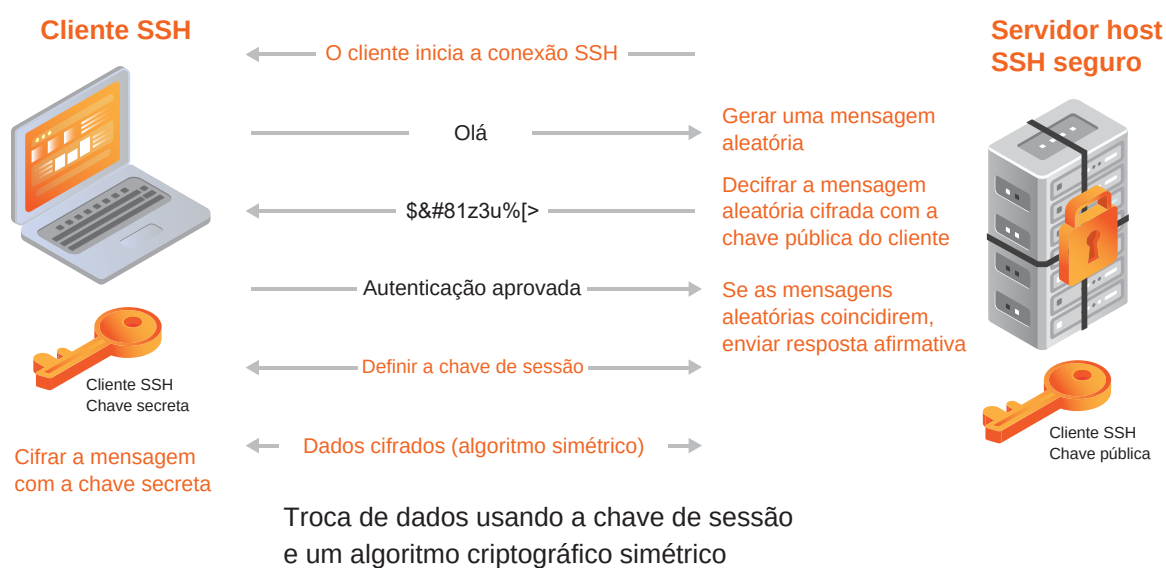


COMO REDUZIR O RISCO DE VIOLAÇÕES DE DADOS?

Empresas experientes em tecnologia conhecem os perigos dos ataques internos e restringem o acesso de seus engenheiros por funções. O controle de acesso baseado em funções (RBAC) limita o acesso dos usuários de acordo com suas funções na organização.

Implementar o RBAC significa que um usuário só tem acesso ao que precisa com frequência para fazer seu trabalho. No entanto, “com frequência” não precisa necessariamente ser diariamente, semanalmente ou mesmo mensalmente. A opção mais segura é conceder aos engenheiros o menor nível de acesso possível. Quando for necessário um acesso de nível mais alto, ele deve ser monitorado, controlado e limitado no tempo.

Como obter um mecanismo de controle de acesso bem monitorado, controlado e limitado no tempo? A melhor solução é um sistema em que cada engenheiro tenha sua própria chave e que permita o acesso a um servidor por um período predefinido — quanto menor, melhor.



QUANDO AS PERMISSÕES DEVEM EXPIRAR?

Normalmente, um engenheiro recebe permissões desde que entra em uma empresa até sair dela. No entanto, uma opção mais dinâmica seria conceder permissões:

- Somente durante um período de plantão
- Até que um ticket do tipo Jira seja fechado
- Somente se a permissão tiver sido usada pelo menos uma vez no último mês

O QUE É UMA CHAVE SSH?

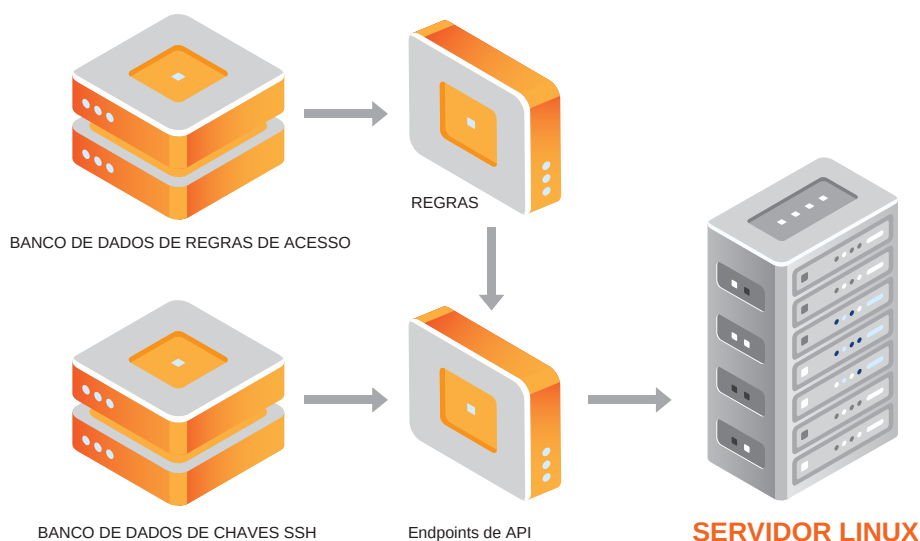
Uma “chave SSH” é um mecanismo criptograficamente seguro para comprovar a identidade de alguém perante um servidor ao fazer login pela rede. Nesse aspecto, ela é semelhante a uma senha longa, mas as semelhanças param por aí. Uma “chave SSH” é, na verdade, composta de duas partes: uma chave privada em um arquivo e uma chave pública em outro. O conteúdo desses dois arquivos é muito diferente, mas eles se complementam e, por isso, são criados ao mesmo tempo.

A metade privada da chave SSH deve ser protegida e nunca, por exemplo, sair do sistema em que foi gerada. Já a metade pública pode ser compartilhada livremente. A chave privada não pode ser recriada a partir da chave pública. Ao copiar a parte pública da chave para qualquer servidor remoto em que esse usuário deva ter permissão de login, o proprietário da metade privada recebe acesso.

COMO CRIAR UM SISTEMA DE GERENCIAMENTO DE CHAVES SSH?

Os componentes necessários para criar um sistema de gerenciamento de chaves SSH são:

-  Um banco de dados que armazena as partes públicas das chaves SSH dos usuários
-  Um banco de regras de acesso que define quais usuários acessam quais servidores
-  Um servidor de aplicativos compatível com aplicativos web simples e endpoints de API



COMO CONFIGURAR:

- 1 Tradicionalmente, o daemon sshd procura no disco o arquivo `authorized_keys` do usuário, no diretório `~/.ssh` dele.
- No entanto, nesta configuração, aproveita-se a diretiva do sshd `AuthorizedKeysCommand` (disponível no OpenSSH a partir da versão 6.2) para instruir o sshd a usar os resultados do script localizado em `~/usr/local/bin/get_keys.sh`. Para isso, altere o local no arquivo `sshd_config`. Observe que, se isso falhar, o sshd ainda recorre ao arquivo `authorized_keys` no disco. Para fazer isso, edite o `sshd_config` assim:

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

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

- 2 O script `~/usr/local/bin/get_keys.sh` pode ser escrito em qualquer linguagem, mas é recomendado usar um script Shell ou Python. O primeiro e único argumento desse script é o nome de usuário Linux do usuário que está solicitando as chaves. O script precisa reenviar a mesma consulta ao endpoint de API criado. Para isso, use o comando `curl`, por exemplo, ``curl https://<endpoint>/get_keys/?user=$1``. Veja abaixo um exemplo do `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 $?
```

Exemplo de solicitação:

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

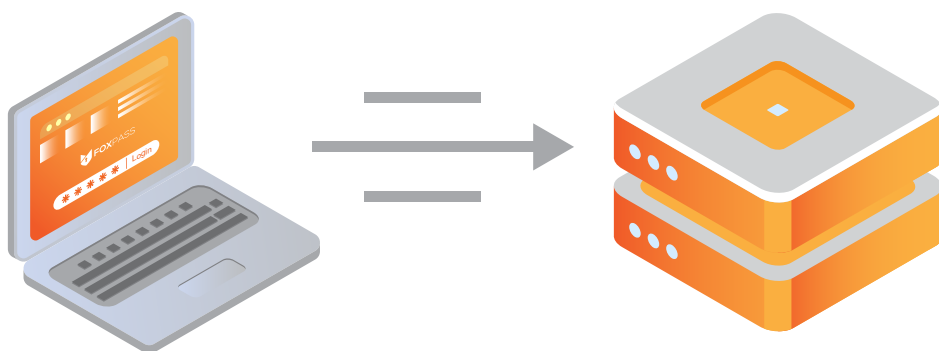
3 Crie o endpoint /sshkeys/ na API de modo que ele:

- obtenha o argumento de nome de usuário da query string;
- procure as chaves SSH desse usuário no banco de dados e
- retorne as chaves SSH no mesmo formato de um arquivo `authorized_keys`: cada chave pública SSH em sua própria linha.

Exemplo:

```
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 Crie um aplicativo web que os usuários possam usar para gerenciar chaves SSH. É fácil autenticar um usuário no aplicativo web usando OAuth do GSuite ou do Office 365. O usuário deve poder adicionar uma nova chave e desativar uma chave antiga.



Interface web simples para login dos usuários (OAuth)
e upload das chaves SSH (metade pública)

BANCO DE DADOS DE CHAVES SSH

Essa configuração básica pode ser ampliada com a introdução de um banco de dados de regras de acesso que regem e controlam o acesso aos servidores. Com a integração a APIs externas, o acesso pode ser modificado com base em condições e reduzido ou removido conforme necessário.

- Para garantir que apenas as chaves SSH dos funcionários atuais sejam distribuídas: fazer uma verificação cruzada com a API do GSuite, Office365, Okta ou OneLogin
- Para impor a rotação de chaves SSH: filtrar as chaves SSH com mais de 90/180/360 dias
- Para restringir quais servidores um usuário pode acessar: verificar os direitos de acesso definidos para cada usuário. Se não existir permissão para o servidor em questão, impedir que o usuário acesse o sistema

Exemplo:

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

`hostname=prod-db-1`

“Máquina de produção” match regex = `^prod-.*`

“aren” tem direitos agora em alguma “Máquina de produção”?

ou

`hostname=prod-db-1`

“Máquina de BD” match regex = `-db-\d+$`

“aren” tem direitos agora em alguma “Máquina de BD”?

CONCLUSÃO

As possibilidades dessa configuração para gerenciamento de chaves SSH e controle de acesso são infinitas. Vulnerabilidades de segurança e ataques internos são sempre uma grande preocupação para organizações com dados sensíveis que podem ser compartilhados com engenheiros, pois um vazamento pode ter impacto enorme. Sempre faz sentido dar aos funcionários o mínimo de privilégios e limitar o acesso a recursos críticos quando concedido. Se o site da Ashley Madison, a Target ou a Tesla tivessem um controle de acesso melhor, talvez não tivessem sofrido violações de segurança tão caras e públicas. O gerenciamento de chaves SSH é um mecanismo útil para proteger a infraestrutura e ajudar a impedir que equipes internas vazem informações privilegiadas para o mundo exterior.

COMO A FOXPASS PODE AJUDAR

A Foxpass oferece produtos completos e fáceis de usar para gerenciamento de chaves SSH e controle de acesso privilegiado, que permitem que os engenheiros acessem os servidores necessários por um período predefinido. Além disso, o Foxpass é um sistema completo de gerenciamento de usuários e grupos para Linux que se sincroniza com diretórios populares como GSuite, Office365 e Okta.