SSH Key Management: Why PAM Tools Fail in Managing SSH Keys?

Privileged Access Management (PAM) tools make the promise of managing SSH keys, but in practice they fail to do so. This leads to a large portion of SSH access remaining unmanaged.

PAM tools apply password management concepts to SSH keys. However, they have limited SSH key discovery and SSH key lifecycle management capabilities and fail to implement SSH Key management best practices.

This white paper explores why this is the case and what the solution is.

Index

Introduction

Today almost every single Unix, Linux, Windows, many IoT systems, and countless network devices come with preinstalled SSH software. Not only is SSH the de-facto way of secure remote administration of servers (on-prem or in the cloud), but it's also extensively used in application integration and automation. Automated SSH usage overwhelmingly uses SSH keys - access tokens fulfilling a similar purpose as passwords, API keys, or other access credentials - as a way to grant access when establishing connections.

However, SSH keys pose a serious problem to effective and comprehensive access management. Even though SSH keys and passwords are both access credentials and need to be managed similarly, they are not the same. SSH keys, due to their long-lasting nature (they don't have an expiration date), can grant access indefinitely if they aren't revoked or removed. Further, if not continuously tracked and managed, keys can accumulate across thousands and thousands of accounts which makes them very hard to locate.

Unmanaged SSH key based access leads to:

- Bypassing existing access controls
- Lateral, and often unauthorized, movement between servers and environments
- Access credentials that never expire
- Keys that have no ownership association
- Scalability problem: Thousands or millions of SSH keys
- Visibility problem: Enabling shadow IT
- Prime targets for attackers

All this makes SSH key management a challenging and demanding task that PAM tools cannot perform reliably, as they are designed to primarily manage passwords, not SSH keys.

We at SSH were the first to offer a scalable, commercial solution to the SSH key management problem that we, as the inventors of the Secure Shell protocol, arguably also created. In fact, all of our enterprise customers already have PAM solutions in place to manage their access, but after evaluating their PAM tools' SSH key management capabilities, they chose to integrate our solution hand in hand with their existing access management to solve the problem that SSH keys pose to their security.

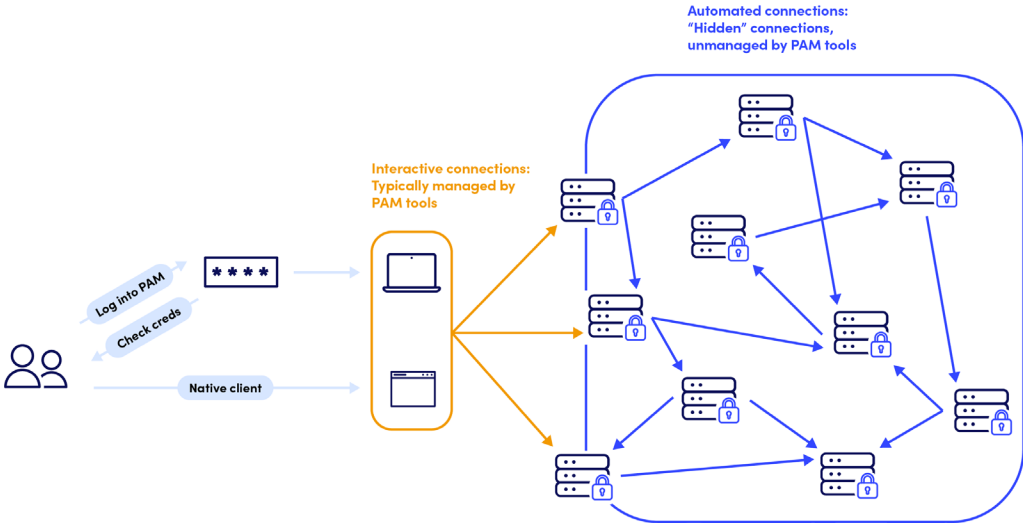
In this white paper, we will discuss the three main reasons why using a PAM tool isn't an effective way to manage SSH access. Additionally, we present a comprehensive solution to the SSH key management problem.

PROBLEM 1: PAM Tools Cover Only 20% of SSH Access

Working with some of the largest enterprises in the world for more than 10 years has revealed some staggering statistics. About 80% of SSH access is comprised of machine-to-machine (M2M) connections, mostly involving automation and service integration systems using non-interactive access with SSH keys.

Often, PAM tools:

- Are not capable of comprehensive discovery and monitoring and leave organizations blind to the actual key sprawl.
- Limit the management of SSH keys to the interactive use case where a user either logs into the target through a jump server or checks out a vaulted private key to use with a native SSH client.
- Are only capable of performing periodic rotation and removal of the interactive access keys (in the image below, this is represented by connections highlighted in orange).



SSH key management using PAM tools: As PAM tools focus only on managing interactive access (highlighted in orange), they lack control over M2M, automated access (highlighted in blue).

PAM tools often fail to understand, analyze, and report trust relationships that exist between servers as a result of application-to-application, automated system access (file transfer, configuration management, and monitoring tools), or other machine-to-machine access requirements. Additionally, machine-to-machine access necessitates non-interactive connectivity - this creates a web of trusts that potentially enables lateral movement from server to server using already provisioned and unprotected SSH keys.

Lateral movement utilizing SSH is an attack vector used by both malicious applications and human actors, including malware, ransomware, as well as state-sponsored and financially motivated threat groups as documented in MITRE's Attack Framework.

PAM solutions provide insufficient control over only a small part of the SSH access. Additionally, they also fail to analyze and report on complex trust relationships created by application-to-application keys. As a result, the risk mitigation related to SSH access achieved by PAM tools is negligible.

PROBLEM 2: PAM Tools Apply the Traditional Password Vaulting Concept to SSH Keys

PAM tools apply their traditional password vaulting concept to SSH keys. This works well for interactive users and connections as it introduces the ability to audit, monitor, and control access that has been traditionally ungoverned for decades. It also introduces Single Sign-On (SSO) and Multi-Factor Authentication (MFA) to access that was not available natively in most SSH protocol implementations.

However, this approach is ill-suited for automated machine-to-machine (M2M) connections which comprise the vast majority of SSH keys.

PAM tools take the traditional concept of vaulting passwords and apply it to SSH keys since both keys and passwords are indeed access credentials. Vaulting may be a viable solution for interactive access albeit outdated and clumsy considering there are PAM solutions offering passwordless and keyless SSH connectivity (more on that later).

Vaulting private keys, which are used for granting SSH access in M2M connections, would require modifying every script and integration relying on SSH keys to include a code that would check out the key from the vault.

A daunting task that adds little additional security in case of private key compromise without proper management of the authorized keys as well.

```
#!/bin/sh
# Monitor disk space and send an email notification
LEVEL=90 # define critical level
EMAIL_USER="itadmin@acme_corp"

ssh -T user@server  << "EOF"
df -H | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{ print $5 " " $1 }' | while read -r output;
do
  echo "$output"
  usep=$(echo "$output" | awk '{ print $1}' | cut -d'%' -f1 )
  partition=$(echo "$output" | awk '{ print $2 }' )
  if [ $usep -ge 90 ]; then
    echo "Running out of space \"$partition ($usep%)\" on $(hostname) as on $(date)" |
    mail -s "Alert: Disk space on $(hostname) above critical level $usep%" "$EMAIL_USER"
  fi
done
EOF
```

Automation scripts rely on SSH access, and sometimes, they utilize SSH keys without explicitly referencing the credential. Vaulting of such keys results in interruption of process flows.

Furthermore, private key vaulting introduces the "chicken-egg problem": merely replacing one credential with another credential, which creates two problems:

- Access to the secret vault is still required from the machine where the private key used to reside before it was placed into vault.
- Credentials for secret vault API calls must be still available to the script/tool trying to establish an SSH connection to perform automated work.

How will this new API credential, which enables authenticated access to the secret vault, be protected and managed then?

Key vaulting is not only infeasible for the majority of SSH access but also the lack of control over the authorized key of a key pair significantly compromises the overall security posture.

A robust key management solution must be able to harden authorized keys, detect access to production environments from unauthorized locations, and report on violations for both interactive and noninteractive access.

PROBLEM 3: Lack of a Future-Proof Strategy in an Ever-Growing SSH Key Landscape

With the expansion of cloud computing, ever more complex automation and application integrations (such as CI/CD pipelines, configuration management, and server monitoring tools), SSH key usage has been steadily increasing over the years. As applications and tools opt to use the convenience and improved security offered by using SSH protocol, large organizations have noticed the demand for provisioning and managing SSH keys steadily increasing, with the number of keys found within their server estates in the millions. As an example, a global financial institution is currently managing 4.5 million SSH keys, while a major retailer is managing 0.9 million SSH keys.

The ever-growing number of SSH keys is a challenge, and PAM tools do not offer key management strategies when the scale is in the millions.

The overwhelming number of SSH keys to manage is not the only issue. While SSH keys are more secure than passwords and extremely useful for automation and application-to-application communications, they are a credential with a standing privilege and no expiration date. That makes keys easy to exploit for unauthorized access. For that reason, SSH keys are a prime target for collection and manipulation to gain access as they maintain persistence in the breached perimeter and achieve privilege elevation.

Not only do PAM tools lack comprehensive key management capabilities, but they also lack a solution to the SSH key expansion problem. A smart solution is to take the path of migration from existing SSH keys to Zero Trust SSH access using just-in-time (JIT) ephemeral certificates.

SSH access with ephemeral certificates is an emerging approach that solves the problem associated with standing privileges, like keys and passwords. Moreover, SSH access with ephemeral certificates that get provisioned just-in-time, following the least privilege paradigm, enables organizations to move towards Zero Trust Architecture.

Finally, the ability to monitor and react to unauthorized provisioning of SSH keys in the server estate remains ever so relevant. While the future of SSH access is certainly keyless and passwordless, at this time, SSH implementations continue to allow the ability for users to self-authorize using their own SSH keys, thus opening the door to bypassing the existing access management solutions.

The Proven Way to Solve the SSH Key Management Problem

In the end, PAM tools not only fail to manage *all* SSH keys, but the key management capabilities offered seem to be an afterthought. Many PAM tools operate with weak or entirely lacking SSH key discovery, management, understanding of key relationships key usage tracking, or policies and reporting.

To manage SSH access effectively, you need a holistic approach that covers the management of both interactive and automated connections. To solve the complex problem of SSH key management you need a scalable and flexible solution with:

A comprehensive SSH key discovery & monitoring

The discovery function allows tracking of SSH access with keys, passwords, and certificates and maps trust relationships in the environment.

A robust policy engine

The solution automatically reports on the state of compliance and can identify high-risk violations, such as unauthorized access to production servers and access with unknown credentials, in addition to the most basic reporting based on key algorithm, size, and age. It can identify unused keys, which can be safely removed. The solution reports on weak ciphers, MACs, and key exchange algorithms used as well as the state of preparedness and adoption of available quantum-safe encryption across the estate.

A wide coverage for often heterogenous server estates

Including mixture of Windows, Linux, Unix, and IBM mainframe z/OS environments. The solution can also identify, report on, and rectify SSH server configurations with insecure settings, such as enabled password authentication for privileged accounts, agent and x11 forwarding, etc.

A scalable API interface and a flexible automation engine

It supports full life cycle management of SSH keys, including audited provisioning of new keys, removal of unnecessary access, hardening existing keys as well as preventing self-provisioning and PAM bypass.

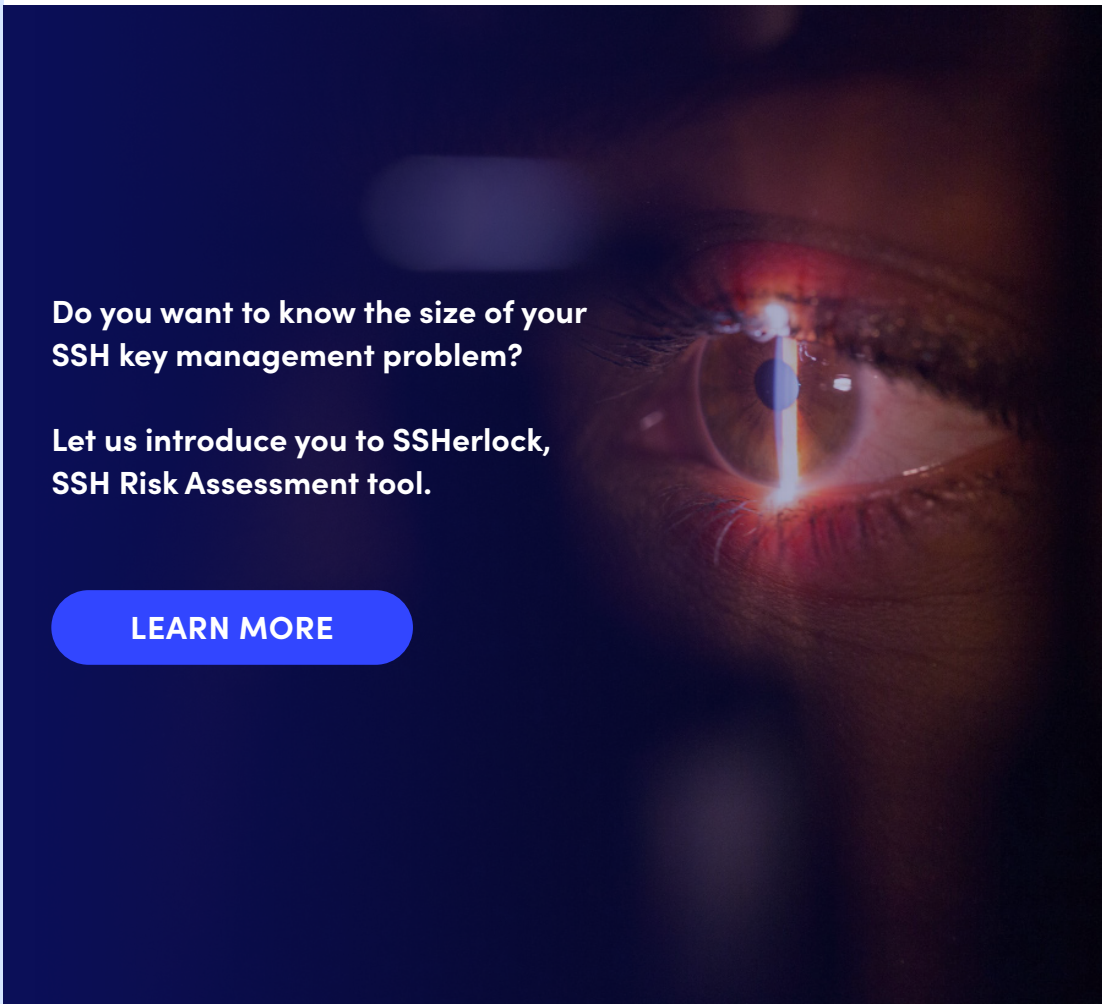
A future-proof path

A path to gradual migration away from perpetual management of standing privileges in the form of passwords and keys.

As the inventors of the Secure Shell protocol, SSH key management is in our DNA. Our PrivX Key Manager offers an all-encompassing approach to SSH key management as well as a migration path towards automated, scalable, and secure SSH access without the need for keys or passwords.

PrivX Key Manager is the only solution on the market that provides the flexibility of a modern, scalable architecture that in addition to utilizing JIT ephemeral certificate access also offers:

- Continuous monitoring for rogue SSH keys
- Full life-cycle management for the keys deemed necessary
- State-of-the-art management and automation capabilities to prevent self-authorizing and allow governing security policies and regulations
- Automatic replacement of existing SSH keys with Just-in-Time ephemeral certificates
- Key migration that is transparent to users and applications to prevent disruption of established processes



SSH

SSH Key Management Using PAM Tools vs SSH's PrivX Key Manager

✓ The method supports the mentioned feature or functionality.

x The method doesn't support the functionality or feature.

– The functionality or feature is limited.

APPROACH	SSH key management with JIT certificates	SSH key management with a PAM tool
EXAMPLE VENDOR OFFERING	**SSH PrivX Key Manager**	**CyberArk, BeyondTrust**
DISCOVERY & POLICY		
User accounts inventory (local, domain, LDAP)	✓	✓
User and host SSH key inventry	✓	–
SSH client/server and configuration inventory	✓	x
Alerting on changes to the environment (rogue keys, unauthorized access, server configurations)	✓	x
Risk assessment via policy evaluations	✓	–
LIFECYCLE MANAGEMENT & AUTOMATION		
Provisioning new key access	✓	–
Key remediation (rotation, restriction, removal)	✓	–
Key relocation to root-owned/central location	✓	x
Future-proof SSH access with JIT ephemeral certificates	✓	x
Migration from existing SSH key access to JIT certificates	✓	x

APPROACH	SSH key management with JIT certificates	SSH key management with a PAM tool
EXAMPLE VENDOR OFFERING	**SSH PrivX Key Manager**	**CyberArk, BeyondTrust**
OPERATIONAL CONSIDERATIONS		
Ease of deployment	✓	x
Bulk operation	✓	x
Delegation of remediation to SSH access owners	✓	x
Accountability and approval processes	✓	x
Addresses SSH key access already in place	✓	-
Reduce key rotation – improve operational efficiencies	✓	x
Role-Based Access Control	✓	-
Session recording	✓	-
Full audit of all source to target connections	✓	x
ICAP integration for virus scanning on file transfers	✓	x
NFS awareness and support in NFS environments	✓	x

We'd love to hear from you!

Get in touch with our experts around the world.

GLOBAL HEADQUARTERS

Helsinki

SSH COMMUNICATIONS
SECURITY CORPORATION
emea.sales@ssh.com

US HEADQUARTERS

New York City

SSH COMMUNICATIONS
SECURITY, INC.
amer.sales@ssh.com

APAC HEADQUARTERS

Singapore

SSH COMMUNICATIONS
SECURITY LTD.
apac.sales@ssh.com

SSH

Let’s get to know each other

Want to find out more about how we safeguard mission-critical access for leading organizations around the world? We’d love to hear from you.

REQUEST A DEMO

SSH