



SSH Key Management Compass - 9 Ways To Manage Your Authentication Keys

The Good, The Bad, & The Ugly

It can be difficult to choose the right key management approach for your organization. This document will reveal the best way to deal with it and point you in the right direction.



Index

Introduction	3
What is the SSH Key Management Problem?	4
SSH Key Management Approaches.....	6
The bad approaches:	6
Switch to using Kerberos	6
Vault authorized keys into a central system (LDAPify)	7
Vault private keys to a central PAM system	7
Key management using FreeIPA.....	8
The ugly approaches:.....	9
Home-grown key management using configuration management tools	9
Migrate to using X.509v3/PKI certificates.....	10
The good approaches:	11
Manage SSH keys with a dedicated solution.....	11
Lockdown authorized keys to a root-owned location locally	12
Migrate to SSH certificates and eliminate SSH keys.....	13
SSH Key Management Best Practices & Standards	14
Discover SSH servers and existing key estate	14
Eliminate at-risk keys	14
Introduce SSH key lifecycle management	14
Migrate to a keyless future	15
Tackle Key Management Today and Tomorrow with SSH.....	15
Approach comparison	17

Introduction

Many organizations aren't aware that there are numerous SSH key management methods available to them. Choosing the wrong approach for your organization, can greatly increase the chances of SSH key mismanagement, potentially leading to inefficient processes, new vulnerabilities, and failed audits.

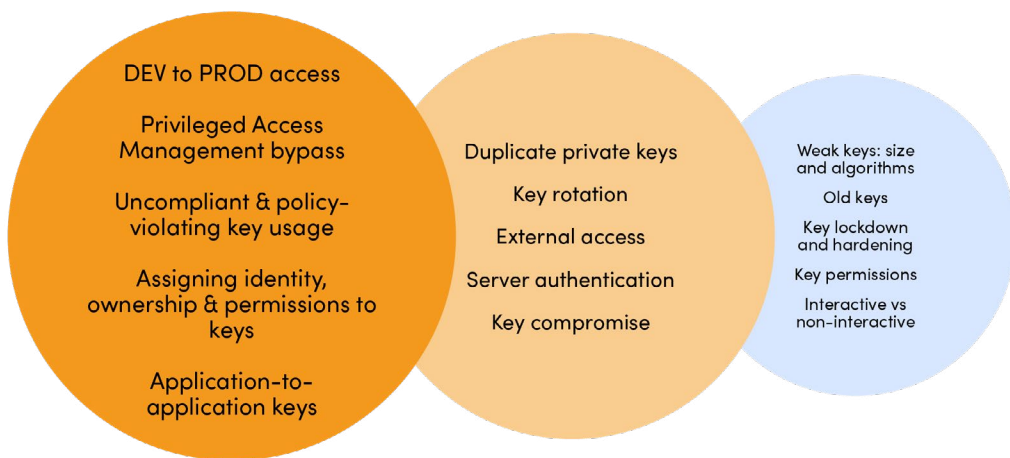
Implementing the wrong approach can additionally incur huge amounts of extra work, and thus, adding new hidden costs to the organization.

On the surface, it might seem that each approach has its own advantages and disadvantages. In reality, there are only a few sensible and effective ways to manage keys thoroughly, at scale, and in an automated fashion.

This guide will demonstrate 9 different ways that organizations all over the world have tried to solve the SSH key management challenge. It will also highlight the selected few methods chosen invariably by some of the most demanding organizations in the world, out of which many are Fortune 500 luminaries.

Learn about the good, the bad, and the ugly ways of managing SSH keys.

Main SSH key management problems to solve include:



SSH key management problems based on urgency to solve them (from the most urgent on the left to the least urgent on the right).

What is the SSH Key Management Problem?

Many credential management and privileged access management (PAM) tools claim to perform SSH key management, but most are unable to do so comprehensively or reliably. This is because most PAM systems leveraged by businesses today are designed primarily to support the governance and control of passwords, but not keys. Of the PAM solutions that do offer key management, many only address user keys in default locations, leaving more complex environments and other SSH keys like server keys (or host keys) vulnerable to compromise.

Why is this the case?

Scalability challenge related to SSH keys, their management, and the chosen management solution



Key duplication related to NFS mounts

Many-to-many relationships

SSH server configuration with non-default locations

NIS+/AD bridging

Tracking key usage

SSH product repackaging

Managing multiple SSH server products

Handling of key options

Operating system support

Managing servers across various time zones

Detection of authorized keys and private keys

Duplicate authorized keys – key material same but options differ

Handling of various SSH authorized key formats

Corrupted key lines inside authorized keys files

Managing files, authorized keys and authorized keys2

Encrypted private key management

Managing unused keys

Tracking of key modifications

Knowledge about what makes SSH key unique

Indirect key activity

Detecting SSH hopping

End users manipulating SSH keys freely

A list of challenges that makes SSH key management difficult (from the most difficult to the least difficult to solve).

SSH keys are notoriously tricky to locate. Since almost anyone can generate traditional SSH keys in an enterprise environment, an end-user is able to save their keys to any location they have access to – anywhere from home directories to enterprise file system areas. Also, solutions that focus on chasing the private keys on the client side may overlook the proper authorization on the server side. Once you've located the keys or inferred their existence from authorizations, the detective work doesn't end there.

Traditional SSH keys have very limited additional data contained within them; this makes it difficult to identify the key's owner just by looking at the key data itself. Many access management solutions focus mainly on interactive user accounts (human users), but might not actively scan and discover non-interactive user accounts. Non-interactive, automated task-oriented user accounts also often have SSH keys under them.

Another issue associated with SSH key management is that traditional SSH keys do not have any expiration dates or revocation capabilities. This means they do not automatically expire and can potentially live forever – supporting access indefinitely – if authorization is not properly managed and removed when no longer needed. Since SSH keys can accumulate into the millions across tens of thousands of accounts, keeping track of every key and maintaining their lifecycles can be an overwhelming task. SSH keys also have another peculiar feature, where users can replicate (copy) private keys and authorized keys across user accounts and/or servers, making discovery and management of duplicate copies a challenge.

For these reasons, 80% of SSH keys go undetected by traditional PAM tools; in the worst cases, organizations may only be appropriately managing 10% of their SSH keys.

Moreover, SSH user keys are not passwords. There are some very significant differences between SSH keys and passwords that require them to be handled differently. There are five key differences to consider:

1. Passwords are related to user accounts. SSH user keys don't have to be.
2. Passwords usually have expiration times. SSH user keys don't.
3. Passwords cannot be generated without oversight. SSH user keys can.
4. Passwords are mostly used for interactive authentication. SSH keys are most frequently used for machine-to-machine authentication.
5. Passwords grant access to the operating system level without additional restrictions. SSH user keys can control both access and privilege levels.

In short, utilizing standard PAM solutions doesn't ensure the complete and thorough management of your SSH keys. It's time for enterprise to consider alternative methods, methods where SSH key management is not a secondary or tertiary add-on function.

SSH Key Management Approaches

The bad approaches:

It is imperative that enterprises enact key management processes regardless of key types in use: whether traditional SSH keys, OpenSSH certificates, or X.509v3 certificates.

Listed below are 9 approaches to SSH key management and their respective benefits and disadvantages.

1 Switch to using Kerberos

Kerberos, a protocol for user authentication between trusted and untrusted hosts and networks, leverages secure access management through mutual authentication. Kerberos uses conventional shared secret cryptography to stop packets in transit from being read, modified, or otherwise attacked.



- Leverages centralized authentication
- SSO capable

- Solves only SSH key problem for user authentication in Secure Shell
- Switching from SSH keys to Kerberos requires a significant overhaul
- Implementation has its own challenges and requires special expertise, especially on Linux/Unix side
- Some machine-to-machine connections may still require the use of keys or tokens
- Promotes vendor lock-in
- Lacks gated access control required for modern-day security needs
- No meaningful logging capabilities for security forensics
- Lacks authorization capabilities, meaning granted access to end resources is often managed on a per-target basis
- Limits usage and visibility of shared accounts, since Kerberos is unable to map their use back to specific entitlements and users

2 Vault authorized keys into a central system (LDAPify)

LDAP (Lightweight Directory Access Protocol) can be used to store user's public key(s) and SSH server customized to obtain the authorization upon authentication from LDAP.



- Effective when installed in completely new server environments that don't contain SSH keys, such as fresh installations in the cloud
 - Leverages centralized management for traditional SSH keys
-
- Requires an active connection to a central LDAP database for each user login
 - Risk of cutting corners with implementation (anonymous bind or LDAP without TLS)
 - High potential for vendor lock-in
 - Mainly for Linux; offers limited OS support
 - Does not support discovery and inventory of existing private and authorized keys on systems
 - Cannot detect changes to the environment – it relies heavily on SIEM to raise alerts when unauthorized connections are made
 - Does not provide an overview of trust relationships created by existing SSH keys
 - Lacks mechanisms to evaluate existing keys against policies such as established security standards and recommendations
 - Difficult to migrate existing key-based connections to certificate-based authentication
 - No built-in reporting capabilities such as obtaining metrics on how the environment is improving

3 Vault private keys to a central PAM system

Traditional PAM tools offer organizations a secure, centralized, and visible platform for storing or vaulting private keys. PAM systems also support streamlined and efficient access management across IT environments, which helps clarify and expedite business operations.



- Leverage the capabilities of an existing solution
- Uses the same logic for vaulting and rotation as for passwords
- End-users cannot modify authorization at will



- Significant vendor lock-in, forcing long-term commitment
- Requires changes to tools and scripts, and often requires the hire of professional services to migrate tools and systems to use private keys from centralized vault
- Relies on an active connection to the enterprise database
- Limited OS support or coverage if endpoint agents are involved
- Depending on the solution chosen, key discovery may be lacking
- Cannot vault all private keys due to source system access blockers – whether they be firewalls, black boxes, or external blockers
- Lacks extensive built-in policies to evaluate the compliance of existing keys against established security standards and recommendations
- Difficult to migrate existing key-based connections to certificate-based authentication
- Anyone can create an SSH key pair anywhere regardless of the private key vault
- Private key vault can become a single point of failure
- Might be missing comprehensive SSH key specific policies and related reporting capabilities
- Might not have capabilities to automate SSH key management tasks such as key deployments, rotations, and automatic key removals



Key management using FreeIPA

Instead of distributing and managing authorized keys and known host files on target systems, SSH keys are uploaded to their corresponding user and host entries in FreeIPA. On FreeIPA-enrolled systems, SSSD can be configured to cache and retrieve user SSH keys, so that applications and services only have to look in a single location for user public keys.



- Can handle management of both user and host keys

The ugly approaches:

- Focused on Linux/Unix
- May become cumbersome in larger environments
- Does not provide discovery or inventory of existing SSH private keys
- Does not provide an overview of trust relationships created by existing SSH keys
- Lacks mechanisms to evaluate existing keys against policies such as established security standards and recommendations
- Difficult to migrate existing key-based connections to certificate-based authentication

5 Home-grown key management using configuration management tools

A number of application and configuration management tools such as Chef, Ansible and Puppet, to name a few, exist to assist IT professionals with automation of server provisioning, application deployment, and various configuration management needs. Those tools can be successfully used to help with SSH key management – however, many of the core features of key management must be implemented separately through scripting (e.g. keys discovery, rotation or detection of unauthorized changes in the environment).



- Simple and powerful way to distribute SSH keys across large estates
- The approach fits in with commonly accepted and widely used existing technology

- Does not provide discovery or inventory of existing SSH private keys
- Cannot detect changes to the environment – it relies heavily on SIEM to raise alerts when unauthorized connections are made
- Does not provide an overview of trust relationships created by existing SSH keys
- Lacks mechanisms to evaluate existing keys against policies such as established security standards and recommendations
- A large amount of development and internal maintenance is required to support this approach

- Difficult to migrate existing key-based connections to certificate-based authentication
- No built-in reporting capabilities such as obtaining metrics on how the environment is improving

6 Migrate to using X.509v3/PKI certificates

X.509v3 certificates are typically used in highly secure or large environments that leverage the widely used X.509v3 public infrastructure (PKI) standard. PKI is seldom deployed only for Secure Shell use cases. This approach is a viable option if PKI already exists in the corporate environment, for example, for TLS use cases or if the government or another third party issued Smart Cards for user authentication. X.509v3 certificates contain a particular user or device identity and can verify public key ownership — effectively eliminating the need for SSH keys.



- X.509v3 certificates carry more data in them, such as expiration date, owner, issuer, and other vital data that can be used for authentication
- X.509v3 certificates are issued by Certificate Authorities in the central PKI system, meaning all certificates in the organization are counted and known — and can be revoked if necessary
- Private key can be stored on HSM, e.g. Smart Card
- SSO capable
- Supported by some solutions, e.g. [Tectia by SSH](#)



- No native support in OpenSSH
- Dependent on larger PKI project and often rigid corporate certificate enrollment practices

The good approaches:



Manage SSH keys with a dedicated solution

Using a dedicated platform to enforce key management allows for a non-intrusive approach to enterprise key management — one that supports automated governance of various key types according to relevant security and compliance standards.



- Can leverage agentless and agent-based SSH server key management
 - Agentless approach can remotely manage keys using SSH protocol on Linux and Unix systems
 - Non-intrusive — no changes or minimal changes required on monitored endpoints
 - No vendor lock-in, and the vendor product is not the central point of failure — that means even if the SSH key manager goes down, existing business connections will continue to function
 - Offers discovery and inventory of existing private and authorized SSH keys
 - Detects and alerts about changes to the environment
 - Offers an overview of trust relationships created by existing SSH keys
 - Evaluates existing keys against policies such as established security standards and recommendations
 - Built-in reporting capabilities such as obtaining metrics on how the environment is improving
 - Some solutions (e. g. UKM by SSH) support the migration from existing key-based connections to certificate-based authentication
 - SSH key management is a primary function of the solution, which can be clearly seen in the solution design: UX layout, available policies, reporting capabilities, scalability, and automation capabilities
-
- Some vendors rely on external online resources such as private key vaults, ephemeral certificate authorities, or static AD/LDAP directories — when this is the case, environment complexity increases, yielding single points of failure or hot-spots



Lockdown authorized keys to a root-owned location locally

Public keys configured and granting access to SSH servers are referred to as authorized keys. SSH servers provide the option to lockdown keys to a privileged or root-owned location instead of the default home directory to prevent the ability of a user to self-authorize access with their own keys. Keys can also be stored, locked, or vaulted in a file system with root-owned permissions or physically on local devices, such as Hardware Security Module (HSM).



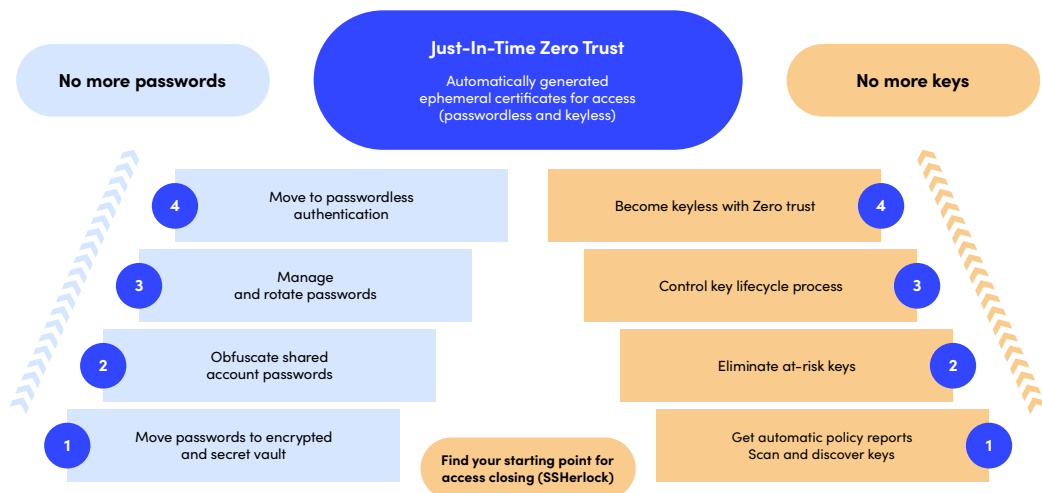
- Recommended by NIST
- No vendor lock-in
- End-users cannot modify their authorized keys because authorized keys (and related directories) are located in a restricted, root-owned location
- Authorized key lockdown allows better change control, as then only the root user is authorized to perform SSH key/access related changes
- This authorized key lockdown approach should be combined with other methods, such as the dedicated solution approach



- Concerned only with limiting the sprawl of authorized keys
- Creates SSH key synchronization problems in clustered services, which then require new processes
- Does not provide discovery or inventory of existing SSH private keys
- Cannot detect changes to the environment – it relies heavily on SIEM to raise alerts when unauthorised connections are made
- Does not provide an overview of trust relationships created by existing SSH keys
- Lacks mechanisms to evaluate existing keys against policies such as established security standards and recommendations
- No built-in reporting capabilities such as obtaining metrics on how the environment is improving
- A large amount of development and internal maintenance is required to support this approach
- Difficult to migrate existing key-based connections to certificate-based authentication

9 Migrate to SSH certificates and eliminate SSH keys

Instead of using permanent access credentials, like SSH keys or passwords, you can switch to temporary access with just-in-time (JIT) certificates that are invisible to the user. SSH certificates also expire by-design, automatically revoking access without administrator or management systems having to clean up keys.



Steps towards keyless and passwordless Just-In-Time Zero Trust access management.



- JIT ephemeral certificates are a major improvement on traditional credentials, including long-term SSH keys or certificates
- Effectively eliminates the need for SSH key management
- Going keyless and passwordless is the future of cybersecurity, and organizations that start migrating their environments now will be ahead of the game – or at the very least, not behind

- Limited to OpenSSH and [Tectia by SSH](#) (version 6.5 and above)
- Not effective in legacy OpenSSH environment, only relatively recent OpenSSH versions
- A dedicated SSH key management solution is often still needed to discover SSH keys and create an inventory from the existing environment, while new SSH connections can be configured to utilize ephemeral SSH certificates; solutions such as UKM can be utilized to convert existing SSH key environment automatically to use SSH ephemeral certificate-based access instead

SSH Key Management Best Practices & Standards

When choosing an SSH key management approach to implement, it is essential to maintain best practices and follow applicable standards. SSH co-authored the NIST IR7966 Document which establishes clear guidelines for SSH key and server management.

Organizations hoping to effectively manage their SSH keys and environments must establish the following management standards and best practices:

Discover SSH servers and existing key estate

Companies that utilize SSH for system administration, automated file transfers, complex integrations between applications, or using configuration and monitoring tools are likely already experiencing SSH key sprawl. [NIST](#) and various regulatory standards highlight the importance of creating inventory of existing SSH keys and trust relationships that they enable.

Enacting continuous key discovery and monitoring allows visibility into your key estate and makes it possible to detect unauthorized SSH access. Applying and validating security policies provides the knowledge necessary to evaluate the risk associated with SSH keys. Automatic reporting allows organizations to remain aware of the state of their security posture and regulatory compliance related to SSH access management.

Eliminate at-risk keys

Establish and apply security policies in order to identify attack vectors that can be eliminated to reduce the risks associated with SSH access. Remove unused keys, replace keys using weak encryption algorithms or too short length sizes, or keys providing unauthorized access to production environments.

Introduce SSH key lifecycle management

It is critical for enterprises to ensure SSH keys and all related accounts and processes are handled appropriately during each phase of their lifecycle. This includes the configuration, distribution, usage, and ultimate termination of SSH keys when necessary.

Leveraging automated tools, processes, and systems can enhance the accuracy and reliability of SSH key tracking, reporting, configuration enforcement, and more. This improves the security and efficiency of SSH key management.

Tackle Key Management Today and Tomorrow with SSH

Migrate to a keyless future

The need for SSH connectivity is likely going to continue to increase and with that the complexity of managing that access. The burden of SSH key management is going to increase as well due to the inherent properties of SSH keys that demand establishing a continuous process of constant clean-up and rotation, much like password management.

Still, SSH protocol implementations provide the ability to utilize Just-In-Time access using ephemeral certificates for establishing trust relationships instead of static credentials, such as keys and passwords.

As the inventors of the Secure Shell, we at SSH are best-equipped to help you enact effective, comprehensive, and long-lasting SSH key management. Our [SSH Zero Trust](#), with its [Universal SSH Key Management \(UKM\) module](#), delivers a holistic and all-encompassing approach to key management that encapsulates all three of the good approaches to SSH key management mentioned above.



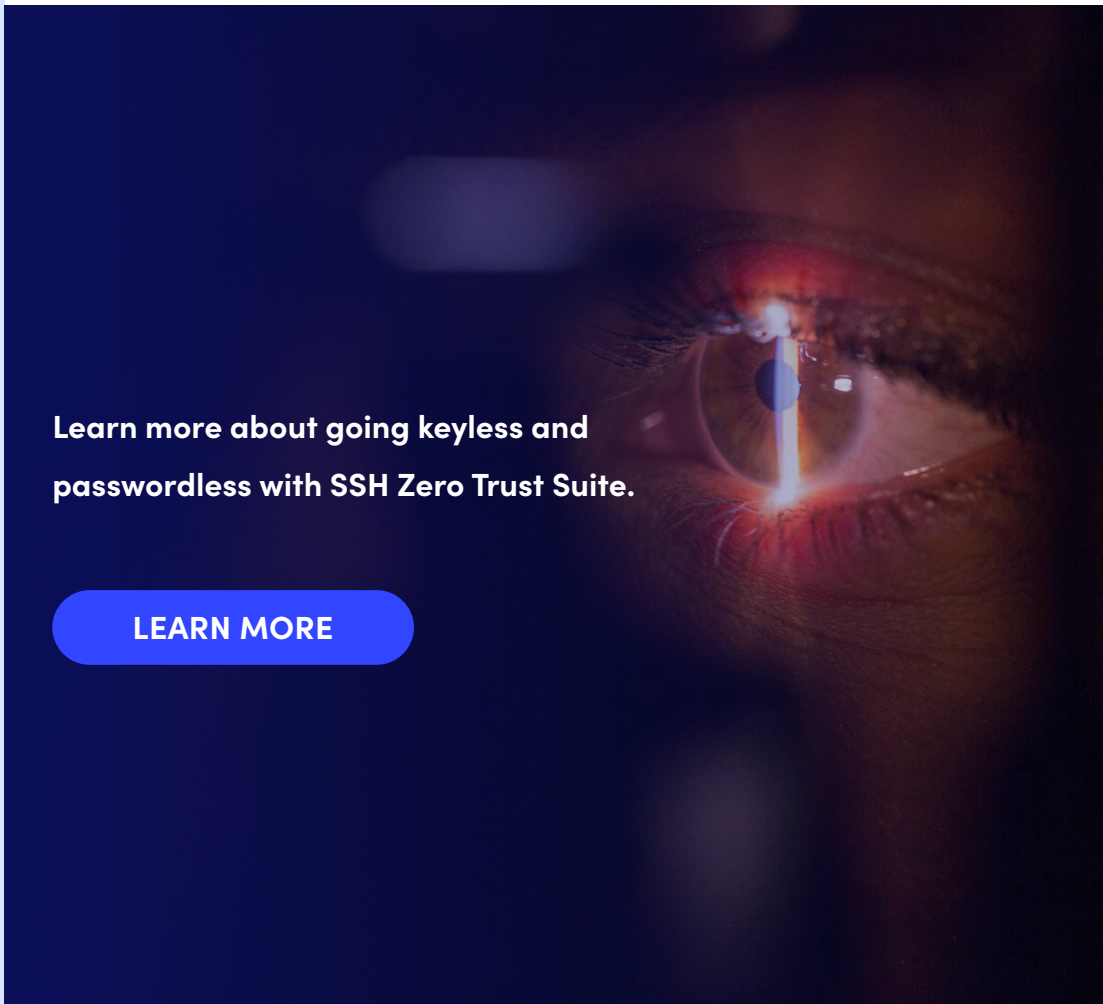
An overview of approaches to SSH key management. A combination of the approaches highlighted in green is the most comprehensive way to manage SSH keys.

We also deliver a path to automatic migration from existing access using SSH keys to scalable and secure use of JIT ephemeral certificates to eliminate the need for management of static credentials and reduce the associated risk.

The UKM module leverages powerful automation, iterative processes, and workflows that support key inventory, discovery, ownership and usage, lifecycle, metadata, accountability, and many more essential key management functions.

Despite the in-depth management capabilities of UKM, no SSH key management solution is foolproof — because all credentials are prone to vulnerability and attack. But what if there were no credentials to manage at all? Enter the world of [keyless \(and passwordless\)](#), where permanent access credentials automatically disappear after use.

With SSH Zero Trust Suite, you can migrate to this future-proof cybersecurity approach while enforcing modern-day SSH key management for your existing credentials. You don't have to switch to keyless right away — instead, you can gradually replace your current SSH keys with just-in-time (JIT) ephemeral certificates for more secure, efficient, and enduring access management.



Learn more about going keyless and passwordless with SSH Zero Trust Suite.

LEARN MORE



Approach comparison

- ✓ The method supports the mentioned feature or functionality.
- ✗ The method doesn't support the functionality or feature.
- The functionality or feature is limited.

APPROACH	Basic authorized SSH key management systems	Comprehensive SSH key management with JIT certificates	Kerberos	Vaulting authorized keys in LDAP	FreeIPA	Vaulting private keys with PAM	Using configuration management tools	PKI controlled SSH x.509 certificates
EXAMPLE VENDOR OFFERING	Venafi, Manage Engine, KeyFactor	SSH Zero Trust Suite (UKM module)	OS native	OS native	Opensource	CyberArk, BeyondTrust	Ansible, Chef, Puppet	N/A
DISCOVERY & POLICY								
User accounts inventory (local, domain, LDAP)	✓	✓	✗	✓	✓	✓	✗	✗
User and host SSH key inventory	✓	✓	✗	—	—	—	✗	✗
SSH client/server and configuration inventory	✓	✓	✗	✗	✗	✗	✗	✗
Alerting on changes to the environment (rogue keys, unauthorized access, server configurations)	✓	✓	✗	✗	✗	✗	✗	✗
Risk assessment via policy evaluations	✓	✓	✗	✗	✗	—	—	✗

APPROACH	Basic authorized SSH key management systems	Comprehensive SSH key management with JIT certificates	Kerberos	Vaulting authorized keys in LDAP	FreeIPA	Vaulting private keys with PAM	Using configuration management tools	PKI controlled SSH x.509 certificates
EXAMPLE VENDOR OFFERING	Venafi, Manage Engine, KeyFactor	SSH Zero Trust Suite (UKM module)	OS native	OS native	Opensource	CyberArk, BeyondTrust	Ansible, Chef, Puppet	N/A
LIFECYCLE MANAGEMENT & AUTOMATION								
Provisioning new key access	✓	✓	✗	—	—	—	—	✗
Key remediation (rotation, restriction, removal)	✓	✓	✗	✗	✗	—	—	✗
Key relocation to root-owned/central location	✗	✓	✗	—	—	✗	—	✗
Future-proof SSH access with JIT ephemeral certificates	✗	✓	✗	✗	✗	✗	—	✗
Migration from existing SSH key access to JIT certificates	✗	✓	✗	✗	✗	✗	✗	✗
OPERATIONAL CONSIDERATIONS								
Ease of deployment	✓	✓	✗	—	—	✗	✗	✗
Bulk operation	✓	✓	✗	✗	—	✗	—	✗

APPROACH	Basic authorized SSH key management systems	Comprehen-sive SSH key management with JIT certifi-cates	Kerberos	Vaulting authorized keys in LDAP	FreeIPA	Vaulting private keys with PAM	Using configuration management tools	PKI controlled SSH x.509 certificates
EXAMPLE VENDOR OFFERING	Venafi, Manage Engine, KeyFactor	SSH Zero Trust Suite (UKM module)	OS native	OS native	Opensource	CyberArk, BeyondTrust	Ansible, Chef, Puppet	N/A
Delegation of re-mediation to SSH access owners	X	✓	X	X	X	X	X	X
Accountability and approval processes	✓	✓	X	X	X	X	X	X
Addresses SSH key access already in place	✓	✓	X	✓	✓	✓	✓	X
Reduce key rotation – improve opera-tional efficiencies	X	✓	X	X	X	X	X	X
Role-Based Access Control	X	✓	X	X	X	—	X	X
Session recording	X	✓	X	X	X	—	X	X
Full audit of all source to target connections	X	✓	X	X	X	X	X	X
ICAP integration for virus scanning on file transfers	X	✓	X	X	X	X	X	X
NFS awareness and support in NFS envi-ronments	X	✓	X	X	X	X	X	X

We'd love to hear from you!

Get in touch with our experts around the world.

GLOBAL HEADQUARTERS

Helsinki

SSH COMMUNICATIONS
SECURITY CORPORATION
Karvaamokuja 2b, Suite 600
FI-00380 Helsinki
Finland
+358 20 500 7000
info.fi@ssh.com

US HEADQUARTERS

New York City

SSH COMMUNICATIONS
SECURITY, INC.
434 W 33rd Street, Suite 842
New York, NY, 10001
USA
Tel: +1 212 319 3191
info.us@ssh.com

APAC HEADQUARTERS

Singapore

SSH CommSec Pte. Ltd.
6 Raffles Boulevard, Marina
Square, #03-308
Singapore 039594
Singapore
Tel. +65 6338 7160
sales.asia@ssh.com

Let's get to know each other

Want to find out more about how we safeguard mission-critical data in transit, in use, and at rest for leading organizations around the world? We'd love to hear from you.

[REQUEST A DEMO](#)