EBOOK

6 Steps to Take Back Control of Your SSH Keys

BEST PRACTICES TO MANAGING SSH KEYS AT SCALE AND MITIGATING THE RISK OF SSH-BASED ATTACKS WITHIN YOUR ORGANIZATION

KEYFACTOR

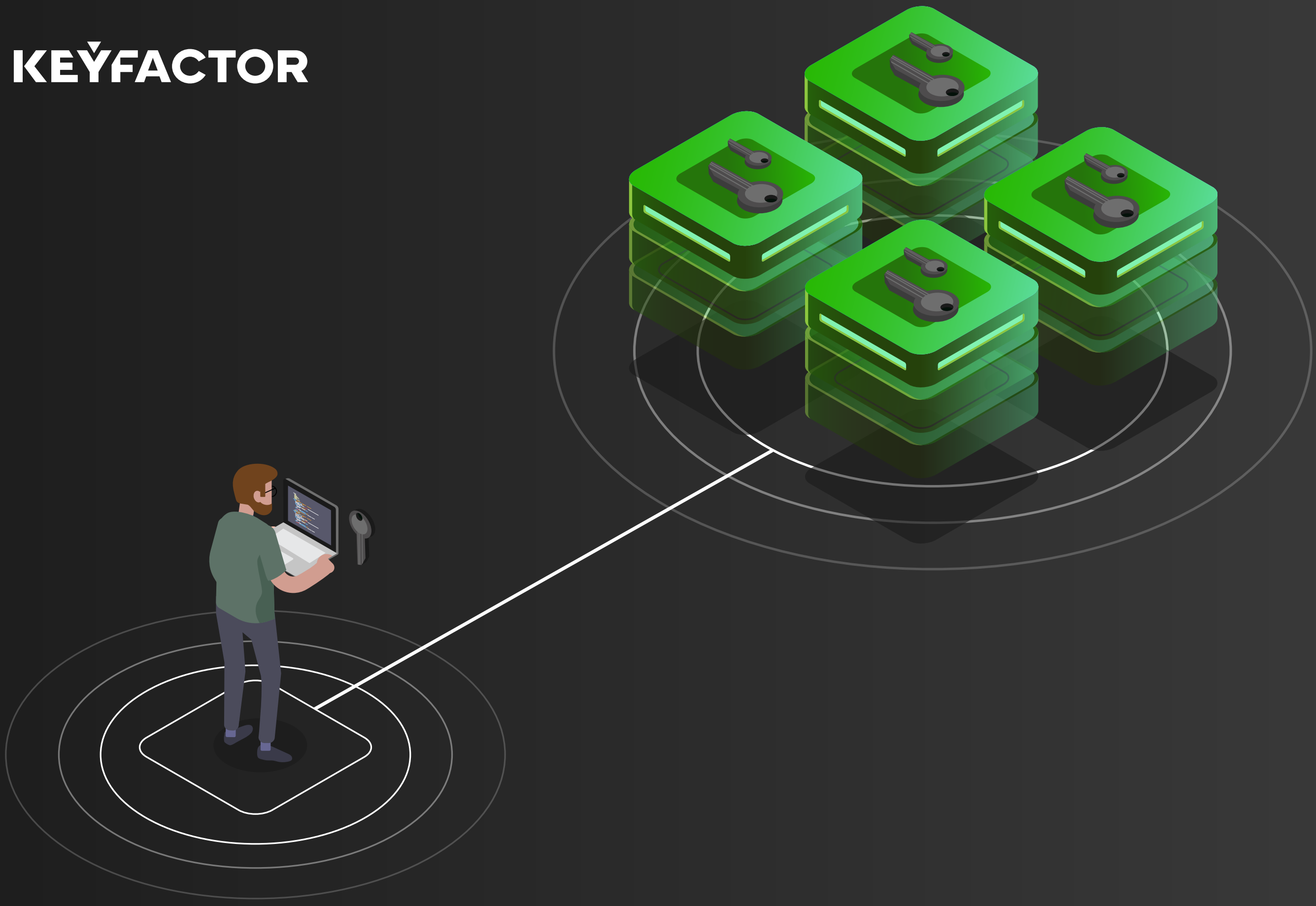


Table of Contents

WHAT'S INSIDE?

This paper provides a brief overview of how Secure Shell (SSH) is used, the risks of unmanaged and unprotected SSH keys, and how to prevent SSH key sprawl.

After reading this paper, you'll be able to:

- **Understand your attack surface** and the causes of SSH key sprawl and excessive privilege
- **Adopt practical steps** to address the risks of unmanaged and poorly configured SSH keys
- **Implement a strategy** to effectively manage, audit, and automate the deployment of SSH keys

INTRODUCTION

SSH keys are one of the most powerful yet overlooked access credentials.

IT and security teams increasingly use SSH to secure remote administrative access and automated processes. As organizations shift to a new IT model driven by cloud and infrastructure automation, the number of privileged SSH connections has proliferated to new levels.

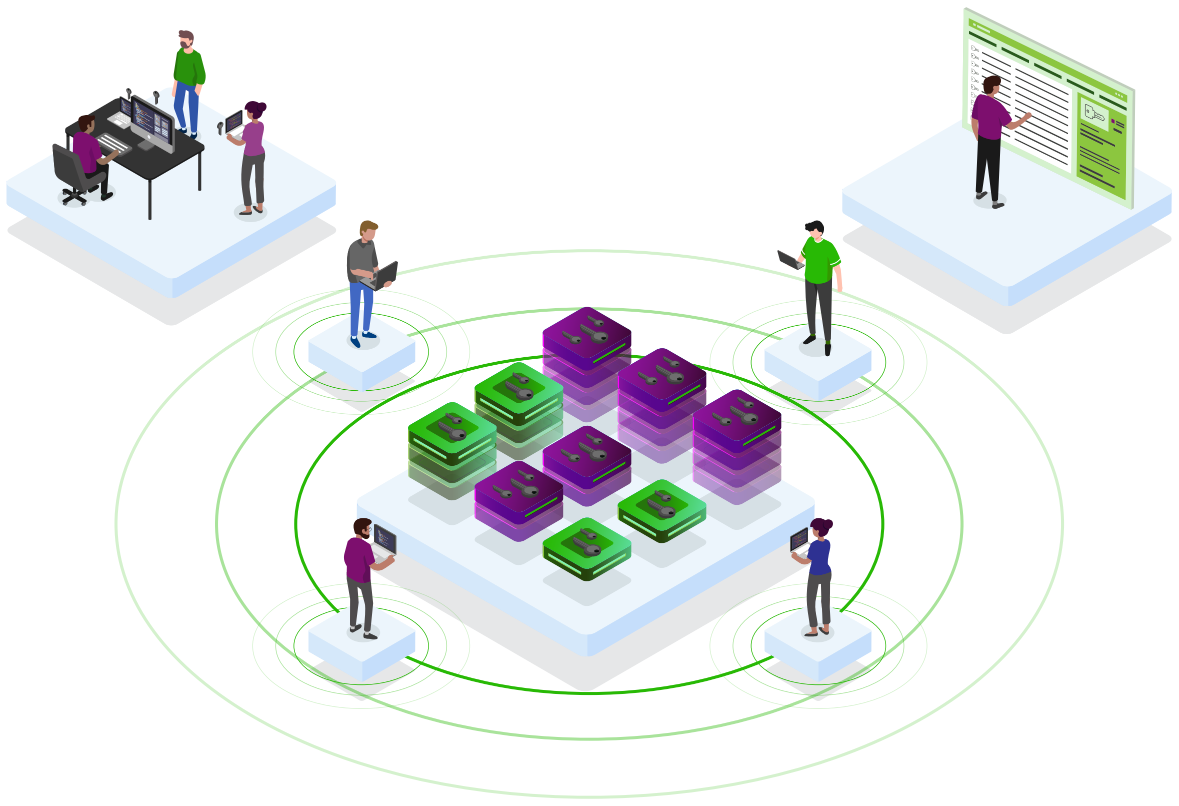
It goes without saying that the volume of SSH keys has exploded. With anywhere from 50-200 keys per server, today it's not uncommon for large enterprises to have upwards of a million keys in their environment.[1]

SSH keys are an incredibly powerful tool. They provide one of the highest levels of trusted access in your infrastructure, yet they typically get less attention from security teams than standard usernames and passwords.

A lack of oversight and governance creates serious risk. Excessive SSH access not only erodes efforts to implement least-privilege or zero-trust initiatives but, also opens the door for hackers to find and abuse the trusted access that SSH keys provide.

The good news is that organizations are getting serious about securely managing keys, certificates, and other secrets across their environment.

Whether you've experienced a breach or you're just looking to better understand the risks caused by unmanaged SSH keys, you're in the right place. In this guide, we'll dig into the root causes of SSH key sprawl and the best practices to take back control of your SSH key management.



[1] https://www.isaca.org/resources/isaca-journal/issues/2017/volume-1/what-every-ciso-must-know-about-ssh-keys

SSH IN A NUT(SHELL)

SSH provides trusted, remote access to systems..

The SSH protocol is a tried and true method to enable secure remote access. It's designed to provide strong, encrypted authentication and communications between users and machines over unsecured networks.

Built into Linux, Unix, and more recently, Windows systems, SSH has become a ubiquitous tool, not just for IT and security teams, but also developers and database admins who use it every day to remotely login to systems.

Why SSH keys vs password logins?

SSH keys are the preferred method of authentication for two reasons: security and ease of use. Unlike passwords, which are susceptible to brute-force attacks, and require users to manually log in to servers, public key authentication is much more secure and doesn't require any user interaction to grant access.

Because public key authentication is non-interactive, users can access a remote server multiple times without needing to manually login every time. This flexibility makes SSH keys ideal for IT process automation, where machines need to securely connect to one another, without any users involved (I.e. secure file transfers, backup and copy, and configuration management tools).

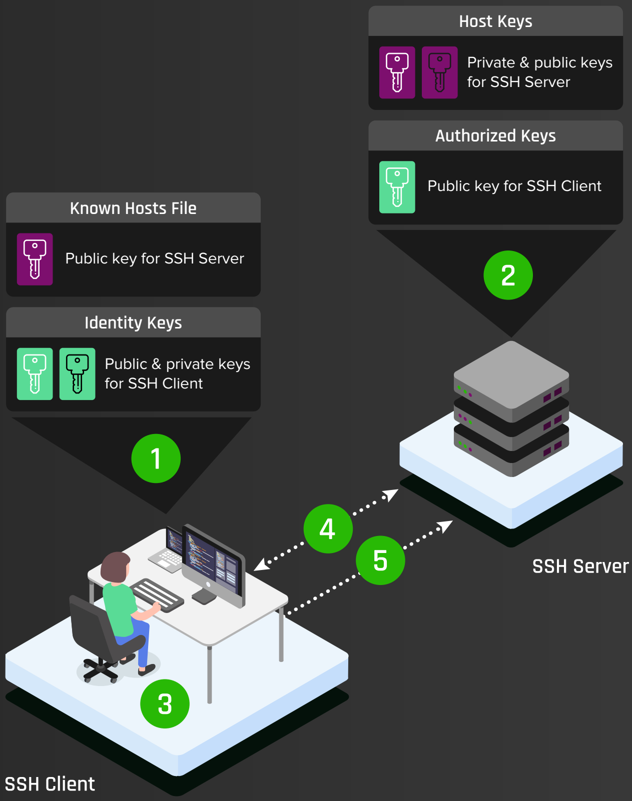
HERE'S HOW IT WORKS

SSH key authentication is easy for users to set up and takes only a few minutes. All they need to do is generate an SSH key pair on their workstation, then share their public key with any server they need to connect to, provided they have the necessary privileges to gain access.

While convenient, handing over carte blanche control over SSH keys to admins that prioritize speed over security, inevitably results in thousands of unmanaged keys.

HOW PUBLIC KEY AUTHENTICATION WORKS:

1. A user or admin generates an SSH key pair on their SSH client, which consists of an identity key (private) and authorized key (public)
2. The user's authorized key is placed in their account on the SSH Server (Host)
3. The user attempts to log in to the remote server
4. SSH client connects to the server and attempts to authenticate the user using their identity key
5. SSH server authenticates the user with their corresponding authorized key


Host Keys
Private & public keys for SSH Server
Authorized Keys
Public key for SSH Client
Known Hosts File
Public key for SSH Server
Identity Keys
Public & private keys for SSH Client
1
2
3
4
5
SSH Server
SSH Client

This image has been abstracted from NIST IR 7966

FOUR ROOT CAUSES OF SSH KEY SPRAWL

Unmanaged SSH keys create excessive levels of privileged access (and risk)..

Despite their widespread use and high-privilege access, most SSH keys are not tightly controlled. Many organizations have built up a large number of untracked keys over time, which makes it incredibly difficult to find and manage them.

Bottom line: the horses have left the barn.

Getting control of the situation isn't impossible, but it starts with understanding how we got here. Here are the most common causes of SSH key sprawl in organizations today:

" There are stunning numbers of SSH keys that grant access – there could be 10-50 times as many as there are usernames and passwords."[2]

01 | UNRESTRICTED USE

If you're like most organizations, you leave it up to your system admins to manage their own SSH keys. Because admins can create and install keys on an ad-hoc basis, or even duplicate or share keys between users, it's easy to lose track of key ownership, access levels, and the relationships between them.

The result is a vast and complex web of many-to-many trust relationships between keys, users, and machines that is next to impossible to audit.

02 | LACK OF GOVERNANCE

System administrators can take certain precautions to restrict the use and access level of authorized keys, but that's more the exception than the rule. For instance, many admins will configure keys for root-level access, even when it isn't necessary.

These types of shortcuts and poor configurations can easily fly under the radar without proper oversight of how and where keys can be used. This creates an attack surface that is many times larger, and more vulnerable, than most security teams realize.

03 | NO EXPIRATION

SSH keys never expire. Unless actively removed, the access they grant is permanent. Just like passwords, keys should be updated regularly and removed when they are no longer needed. Often, though, this is not the case.

If an employee leaves the organization or an automated process is taken down, related keys often end up unaccounted for in various files that can still be accessed by an unauthorized user – whether a malicious insider or an outside attacker.

04 | HARD TO REMEDIATE

In many environments, up to 90% of SSH keys are unused or unmanaged.[3] Security teams may even be aware of this, and the attack surface it creates, but removing even one key can be risky and a logistical nightmare.

Without any insight, they simply don't know which keys are still active, who owns them, or what the keys are being used for. If an active key is removed accidentally, they risk breaking a system or causing a widespread outage.

[2] https://www.isaca.org/resources/isaca-journal/issues/2017/volume-1/what-every-ciso-must-know-about-ssh-keys
[3] https://www.csoonline.com/article/3196974/unmanaged-orphaned-ssh-keys-remain-a-serious-enterprise-risks.html

THE NEW NORM: CLOUD, DEVOPS & WFH

SSH isn't going anywhere. In fact, it's everywhere.

Cloud-first, agile development, and work-from-home strategies are ramping up, making SSH even more critical for IT and security teams. Meanwhile, the invisible line we draw between what is trusted and what is not has blurred. As organizations strive to adopt a "zero trust" security model, uncontrolled sprawl of SSH keys and the access they grant creates a serious problem.

SSH IN THE CLOUD

Linux dominates the cloud. Nearly 90% of all public cloud workloads run on Linux.[4] As the de facto credential to secure remote access for Linux (and Unix-like) machines, SSH has consequently taken off.

SSH is inherently secure, but exposing access, or worse, credentials to the entire internet opens the door to attackers. Despite this well-known risk, 32% of internet-exposed hosts in the public cloud have open SSH services.[5]

Even worse, a 2020 report by Orca Security found that nearly 6% of internet-facing assets contain SSH keys that could be used to access adjacent systems.[6] If left unaddressed, these vulnerabilities will undoubtedly be exploited by attackers.

SSH IN DEVOPS

Developers rarely needed to use SSH in their day-to-day work before the rise of DevOps. Now SSH has become a fundamental security tool for IT Ops and developers use to collaborate more effectively in automated build and release processes.

Whether it's developers connecting to Git repos, running scripts to connect to remote workloads, or configuration management tools such as Chef, Puppet, and Ansible that remotely access servers to make local changes, SSH is everywhere.

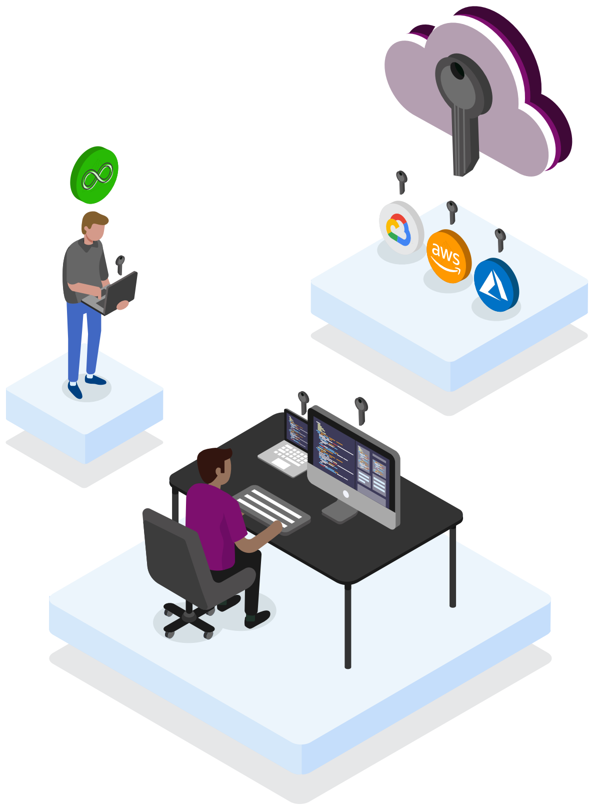
However, both interactive (user) and automated SSH key-based access must be controlled. Giving developers free reign to create and copy SSH keys is a huge security risk. The right approach requires a balance between security and efficient access.

SSH FOR WFH

Work-from-home (WFH) experienced a surge following the spread of COVID-19. Even as the world returns to normal, the remote workforce isn't about to reverse course anytime soon.

Because SSH allows users to configure and manage systems from anywhere, there's no doubt it will be used even more extensively. What you don't want is for your remote management strategy to turn into a risk liability.

Having regular visibility into how SSH keys are being used, who owns them, and the level of access they provide will be essential for security and risk leaders moving forward.



[4] https://www.developer.com/daily_news/90-of-the-public-cloud-runs-on-linux.html
[5] https://unit42.paloaltonetworks.com/hunting-the-public-cloud-for-exposed-hosts-and-misconfigurations/
[6] https://www.businesswire.com/news/home/20200728005324/en/Orca-Security-Research-Finds-Public-Cloud-Environments

SSH-BASED ATTACKS ON THE RISE

As the de facto credential for remote access, SSH keys are a natural target.

SSH is secure – until it's not. While the protocol is inherently trusted, it's only as secure as the keys on both ends. Using a stolen SSH key, an attacker can gain access to a server, search for more keys, and then access more servers via SSH to move laterally and spread their attack surface. It all starts with just one key.

We know that hackers have been after SSH keys for years, but the frequency and sophistication of their attacks have grown. Below are just some examples of recent threats and attacks that have surfaced.

HOW DO ATTACKERS EXPLOIT SSH VULNERABILITIES?

Brute-force attacks compromise SSH password logins

Advanced malware attempts to find & extract SSH keys

Attackers pivot between systems & move laterally within networks

Attackers insert authorized keys to create a backdoor

- **Azure (2019):** Microsoft disclosed a Linux-based cyberattack where an attacker brute-forced a password-based login for SSH to launch large-scale attacks.[7]
- **Return of the WIZard (2019):** Qualys uncovered a pervasive worm that exploits vulnerable Linux email servers and installs authorized keys into the authorized_keys file to create SSH backdoor access.[8]
- **Golang malware (2019):** Researchers at F5 discovered a cryptominer campaign that targeted Linux-based servers. One of the malware propagation techniques was attempting to connect to and infect other machines using found SSH keys.[9]
- **FritzFrog (2020):** Security firm Guardicore discovered FritzFrog, an active botnet campaign which uses an SSH brute-force attack to infiltrate SSH servers, collect found SSH keys to target other hosts, then install its own authorized key to create persistent backdoor access into the infected machine.[10]
- **Lemon_Duck (2020):** Sophos labs uncovered an advanced cryptojacker that starts with an SSH brute force attack with username 'root' and a hardcoded list of passwords. If the attack is successful, it launches a malicious shellcode to look for additional targets in /.ssh/known_hosts to spread without detection.[11]

[7] https://www.microsoft.com/security/blog/2019/05/23/uncovering-linux-based-cyberattack-using-azure-security-center/
[8] https://www.cybereason.com/blog/new-pervasive-worm-exploiting-linux-exim-server-vulnerability
[9] https://www.f5.com/labs/articles/threat-intelligence/new-golang-malware-is-spreading-via-multiple-exploits-to-mine-mo
[10] https://www.guardicore.com/2020/08/fritzfrog-p2p-botnet-infects-ssh-servers/
[11] https://news.sophos.com/en-us/2020/08/25/lemon_duck-cryptominer-targets-cloud-apps-linux/

WHY IT'S TIME FOR A NEW APPROACH

Traditional SSH key management isn't working. Here's why.

If it wasn't clear before you started reading this paper, it's evident now that our security community has a serious SSH problem on its hands.

Not only are SSH-based attacks and malware propagation disruptive to security, they're also near impossible to detect, since they typically spread through encrypted channels.

However, with the right tools and processes in place to find and manage SSH keys across your infrastructure, you can more effectively respond to and even prevent many of these attacks. Just as importantly, you can simplify the task of complying with audits and least-privilege requirements.

Why traditional, manual methods don't work.

Traditional approaches to SSH key management aren't just labor-intensive, they're also prone to error and minimal oversight. Between admins sharing keys and a lack of rotation and termination policies, the risk of SSH key exposure grows with each passing day.

You could manually log into each machine to scan authorized keys, but this just isn't practical, even in modestly complex environments.

More users, more servers, and more keys create hours of tedious work to keep policies and access controls in check. As workloads are spun up and torn down, or employees come and go, it becomes impossible to keep an accurate and up-to-date inventory.

Next, you have the issue of unused keys, unnecessary root keys, and weak or outdated keys. Manually rotating or deleting keys on target systems is a daunting task.

Without proper oversight into all of these keys and their trust relationships, it's far too easy to miscalculate and accidentally remove an active key used for critical access.

All of that said, you can't simply stop system admins from using SSH keys. They are far too important in day-to-day IT and security operations. A successful SSH implementation should automate key provisioning and updates, while ensuring proper policy guardrails are in place to maintain control.

HERE'S WHAT YOU NEED TO MANAGE KEYS EFFECTIVELY:

Discovery tools to find keys & map trust relationships

Monitoring & reporting capabilities to identify risks

Centeralized governance over key rotation & usage

Automation and self-service to provision & update keys

6 STEPS TO TAKE BACK CONTROL

SSH key management is built on process, enabled by the right tools.

If the horses have left the barn, it's time to build a better barn.

Finding and eliminating existing vulnerabilities is important, but SSH key management should primarily focus on creating a new framework for how keys should be deployed and managed. That means not just removing existing SSH keys that pose a risk but, starting fresh by deploying new SSH key pairs using a centralized, administered process.

1 DISCOVER & MAP KEYS

The first step in eliminating SSH key sprawl is to discover existing keys within your network and bring them into a centralized repository. Once you've gathered all of your keys, you can start to map key-user relationships to better understand what they grant access to.

Using a network-based mechanism to discover keys takes the heavy-lifting out of inventory and mapping trust relationships to associated users (private keys), servers and service accounts.

WHAT YOU NEED:

- Agentless or agent-based discovery of SSH keys on remote servers and hosts, instead of manual tracking
- A searchable inventory of all SSH keys – including data such as key rotation status, key type and access level
- Trust mapping capabilities to understand trust relationships between key pairs, users, and machines

2 ANALYZE YOUR RISK

Once discovered, SSH keys must be thoroughly analyzed for potential vulnerabilities, either in configuration or usage. For instance, you'll want to make sure that keys are only configured for root-level access when it's necessary. This phase is all about finding ways to reduce your potential risk exposure and achieve better 'crypto-hygiene' for auditability. Here are some common key-related vulnerabilities:

ROOT ACCESS

In some cases, authorized keys are configured to grant root-level access. If root keys are compromised, an attacker essentially has free reign to modify, infect or even destroy the host.

FORGOTTEN KEYS

Keys created by admins for "temporary" use are often forgotten when they leave the company or no longer need access to that system. Attackers can leverage these keys to move undetected between systems.

ORPHANED KEYS

Orphaned keys are authorized (public) keys without a known private key. If the location and security of the private key is unknown, there is no way of knowing if it has been compromised.

WEAK KEYS

Shorter key lengths make it easier for malicious actors to 'crack' SSH keys. When (not if) algorithms are deprecated, it's important that you can replace any affected keys.

3 REMEDIATE VULNERABILITIES

Once you've identified your risk exposure, you can take action to reduce it. This isn't a one-time effort though – continual monitoring and reporting are essential in identifying new risks as they surface, such as rogue keys created out-of-band.

With a complete and accurate inventory of all SSH key pairs, you can start to rotate or replace weak and outdated keys, remove duplicate keys or keys with unnecessary root access, and clean up unused or orphaned keys. If someone leaves the company, you can remove their keys from your servers to keep things clean and secure.

WHAT YOU NEED:

- Continuous monitoring of authorized key files on Linux or Unix machines
- Automated alerts when rogue keys are created
- Drill-down reports to regularly identify and remediate new risks
- Automated key provisioning to add, delete, or rotate keys on remote servers

4 CREATE FRESH KEY PAIRS & ROTATE THEM REGULARLY

The best practice approach is to delete all untracked keys and replace them with freshly generated key pairs. Establish a streamlined process that allows specific authorized users to easily create and deploy keys through a simple, yet controlled workflow.

Once new key pairs have been generated, they should be rotated regularly at pre-defined intervals to maintain compliance with internal or external policies and reduce the additional risk exposure that 'stale' keys create.

Whether key rotation is triggered by the user via UI/API or automatically by the system (forced rotation), the backend process of provisioning new keys and removing the old keys on remote servers should be automated.

WHAT YOU NEED:

- Self-service UI or API for SSH users to generate and rotate keys
- Automated alerts to notify users when to rotate keys (if manual)
- Integration with users and groups from Active Directory (AD) or another identity provider
- Configurable maximum key lifespans

5 CONTROL SSH KEYS & ACCESS

Now that you've deployed fresh SSH key pairs to target systems, it's important to define permissions for each key and control who has SSH-based access to which systems.

This can really only be achieved with a proper SSH key management tool, which allows you to centrally assign or revoke access to SSH hosts based on specific users and groups, while orchestrating keys in the backend to facilitate those controls.

WHAT YOU NEED:

- Integration with AD or your identity provider to tie users to associated SSH logons
- Agent-based or agentless key management for remote systems
- Role-based access controls for key owners, server admins, and security admins

6 CONTINUOUSLY MONITOR

Unknown SSH keys pose a continuous threat to your organization. Achieving 100% control isn't possible, but it is possible to stay ahead of threats by regularly conducting audits and continually monitoring your key inventory.

It also helps to maintain an ongoing audit log of important events, such as key rotation, generation and provisioning.

WHAT YOU NEED:

- At-a-glance compliance dashboards with drill-down capabilities
- Scheduled reports, such as reports on SSH key types, rotation status, and access privileges
- Comprehensive audit logs of all user and key lifecycle-related activities

SUMMARY

Putting a stop to SSH key sprawl starts here. By following these six steps, and adopting the right tools and processes, you can start to take back control of SSH keys and the trusted access they grant within your organization.

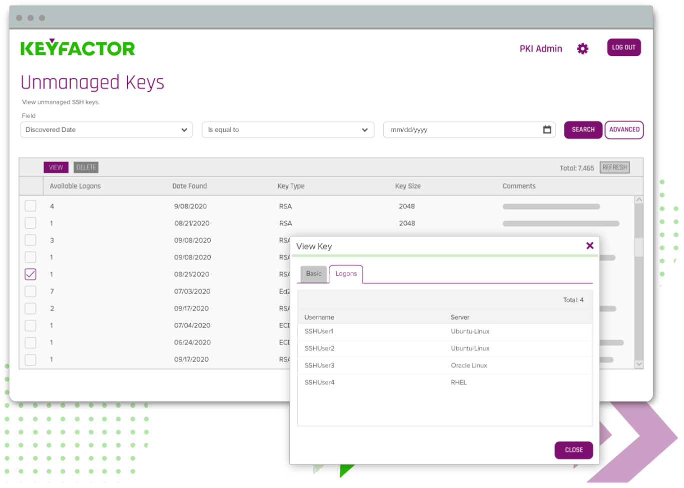
KEYFACTOR

Next Steps

Find out how the SSH Key Manager for Keyfactor Command can enable you to effectively manage and control SSH keys across your on-prem and cloud environments.

Request a demo of Keyfactor Command today.

Request a Demo ▶



Unmanaged Keys

View unmanaged SSH keys.

Total: 7,465

	Available Logons	Date Found	Key Type	Key Size	Comments
	4	9/08/2020	RSA	2048	
	1	08/21/2020	RSA	2048	
	3	09/08/2020	RS		
	1	09/08/2020	RS		
✓	1	08/21/2020	RS		
	7	07/03/2020	Ed2		
	2	09/17/2020	RS		
	1	07/04/2020	EC		
	1	06/24/2020	EC		
	1	09/17/2020	RS		

View Key

Total: 4

Username	Server
SSHUser1	Ubuntu-Linux
SSHUser2	Ubuntu-Linux
SSHUser3	Oracle Linux
SSHUser4	RHEL

KEYFACTOR

Keyfactor empowers enterprises of all sizes to escape the impact that breaches, outages and failed audits from mismanaged digital certificates and keys have on brand loyalty and the bottom line. Powered by an award-winning PKI as-a-service platform for certificate lifecycle automation and IoT device security, IT and InfoSec teams can easily manage digital certificates and keys. And product teams can build IoT devices with crypto-agility and at massive scale. Exceptional products and a white-glove customer experience for its 500+ global customers have earned Keyfactor a 98.5% retention rate and a 99% support satisfaction rate.

Learn more at **keyfactor.com**

CONTACT US

- www.keyfactor.com
- +1.216.785.2990