Aug. 21, 2014

NIST IR 7966

DRAFT Security of Automated Access Management Using Secure Shell (SSH)

NIST announces the public comment release of Draft Interagency Report (IR) 7966, ***Security of Automated Access Management Using Secure Shell (SSH)***. The purpose of this document is to assist organizations in understanding the basics of Secure Shell (SSH) and SSH automated access management in an enterprise, focusing on the management of SSH access tokens. It discusses the basics of access management and automated access management and it examines the basics of SSH version 2.0. It describes the primary categories of vulnerabilities in SSH user key management and recommends possible mitigations for each category of vulnerability then it lists recommended practices for management. It explains risk mitigation for SSH access tokens. and it concludes with solution planning and deployment.

Please send your comments to NISTIR7966-comments @nist.gov by ***September 26, 2014*** using the following template.

NISTIR 7966 (Draft)

Security of Automated Access Management Using Secure Shell (SSH)

Tatu Ylonen
Karen Scarfone
Murugiah Souppaya

National Institute of
Standards and Technology
U.S. Department of Commerce

NISTIR 7966 (Draft)

Security of Automated Access Management Using Secure Shell (SSH)

Tatu Ylonen
SSH Communications Security
Helsinki, Finland

Karen Scarfone
Scarfone Cybersecurity
Clifton, Virginia

Murugiah Souppaya
Computer Security Division
Information Technology Laboratory

August 2014

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Acting Under Secretary of Commerce for Standards and Technology and Acting Director

National Institute of Standards and Technology Interagency or Internal Report 7966
43 pages (August 2014)

Public comment period: *August 25, 2014* through *September 26, 2014*

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: NISTIR7966-comments@nist.gov

Reports on Computer Systems Technology

Abstract

Hosts must be able to access other hosts in an automated fashion, often with very high privileges, for a variety of reasons, including file transfers, disaster recovery, privileged access management, software and patch management, and dynamic cloud provisioning. This is often accomplished using the Secure Shell (SSH) protocol. The SSH protocol supports several mechanisms for authentication, with public key authentication being recommended for automated access with SSH. Management of automated access requires proper provisioning, termination, and monitoring processes, just as interactive access by normal users does. However, the security of SSH-based automated access has been largely ignored to date. This publication assists organizations in understanding the basics of SSH automated access management in an enterprise, focusing on the management of SSH access tokens.

Keywords

access control; authentication; automated access management; device authentication; Secure Shell (SSH); user authentication

Acknowledgments

The authors wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content.

Trademark Information

Table of Contents

1. Introduction

1.1 Purpose and Scope

The purpose of this document is to assist organizations in understanding the basics of Secure Shell (SSH) and SSH automated access management in an enterprise, focusing on the management of SSH access tokens.

1.2 Audience

This document will be created for security managers, engineers, administrators, and others who are responsible for acquiring, testing, implementing, and maintaining SSH solutions involving automated access management. Portions of the document may be of interest to SSH users.

1.3 Document Structure

The remainder of this document is organized into the following sections and appendices:

- Section 2 discusses the basics of access management and automated access management.
- Section 3 examines the basics of SSH version 2.0.
- Section 4 describes the primary categories of vulnerabilities in SSH user key management and recommends possible mitigations for each category of vulnerability.
- Section 5 lists recommended practices for management.
- Section 6 explains risk mitigation for SSH access tokens.
- Section 7 discusses solution planning and deployment.
- Appendix A provides a mapping to NIST SP 800-53 security controls.
- Appendix B provides a mapping to Cybersecurity Framework subcategories.
- Appendix C lists tools related to SSH and SSH automated access management.
- Appendix D contains a detailed discussion of a particular problem that is relevant across a variety of industry sectors.
- Appendix E defines selected acronyms and abbreviations used in the document.
- Appendix F defines selected terms used in the document.
- Appendix G lists resources.

2. The Basics of Access Management and Automated Access Management

Controlling access to information systems is critical for information security. Access controls exist on many levels and using many technologies. The levels include physical restrictions on access to hardware; logical controls for accessing network interfaces, hardware management ports on servers, virtualization hypervisors, operating system (OS) user accounts, and information through database systems; and logical controls implemented by applications.

Information security (confidentiality, integrity, and availability) is compromised if controls at any of these levels fail. Breaches at different levels have different implications. Generally, a breach at the hardware, hypervisor, OS, or database level is more serious than a breach at application level. For example, breaking into a database account on a server may permit reading, modifying, and destroying any data in a database, bypassing normal database-level controls. Breaking into a "root" (administrator) account generally permits doing this to all data on all accounts on the system, plus installing deeply hidden backdoors, modifying the operating system, corrupting data, or rendering the system unbootable.

Most operating systems use user accounts as the primary unit of access control. In this document, a user account means an OS level user account unless otherwise specified. Many user accounts correspond to people (including system administrators), while service accounts are used for running application software or are used internally by the OS. It is also worth noting that many applications implement their own user accounts that do not correspond to OS level user accounts (they essentially share the service account of the application); however, such application-level accounts are generally beyond the scope of this document.

User accounts may be stored in a centralized repository (e.g., Active Directory [AD] or Lightweight Directory Access Protocol [LDAP]) or may be configured locally on a system. A user account defined locally is generally distinct on each system (separate password, separate home directory, etc.), even if the same account name is used on multiple computers, while user accounts defined in a centralized repository are often available on more than one computer and share the same home directory (on a networked file system). Service accounts are very commonly local accounts, and accounts for people are often stored in a directory.

In a very real sense, access control is the essence of information security. Other security technologies primarily exist to implement and enforce access controls, to make it harder to analyze and attack access control systems, limit the impact of actual breaches, evaluate the current state of protections, detect suspicious activity, counteract undesired activity, or help analyze what happened after the fact. The critical balance in information security is between the need to grant access and the need to limit access. Consequently, access must be provisioned (based on proper justification for that level of access) and it must be eventually terminated (e.g., when an employee leaves the role that justified the access, when a client system is decommissioned).

Access has become increasingly automated. Examples of this automation include file transfers, disaster recovery, privileged access management, software and patch management, and dynamic cloud provisioning. This automation involves transferring data and executing commands, such as having hosts reconfigure other hosts. Thus, hosts must be able to access other hosts, often with very high privileges. Unfortunately, there has been little planning and oversight of automated machine-to-machine access control. Instead, such access has been added and configured on an ad hoc basis by system administrators, vendors, and integrators as part of other projects, without formal access control lifecycle management (e.g., standardized provisioning and termination processes, access token management (e.g., periodic password changes)). This publication explores the field of SSH-based automated access management, with a strong focus on security issues and how to best address them.

3. The Basics of SSH

This section examines the basics of SSH version 2.0. First, Section 3.1 explores SSH protocol basics. Next, Section 3.2 discusses the most common use cases for SSH, including the use case scenario of interest in this publication: automated access. Finally, Section 3.3 discusses the user authentication mechanisms that SSH supports and how they relate to automated access.

3.1 Protocol Basics

Secure Shell (SSH) is a protocol for logging into a remote host and executing commands on that host (e.g., administrative commands). SSH software is available for nearly every platform, and SSH is also embedded behind the scenes into a wide variety of IT, networking, and security technologies, including file transfer, systems management, identity management, and privileged access management. To summarize, the SSH protocol is widely used for remotely connecting to hosts, integrating hosts, and automating their operation.

What distinguishes the SSH protocol from earlier remote administration protocols, such as telnet, remote shell (rsh), remote login (rlogin), and remote copy (rcp), is its built-in support for robust security features such as user authentication, device authentication, and transmission encryption. SSH has almost completely taken the place of these insecure remote administration protocols.

The SSH protocol has a typical client/server architecture. An SSH client application on host A initiates a connection to an SSH server application on host B. These two hosts negotiate encryption for their transmissions, then perform device authentication for the server host (host B)[1], and finally send client authentication credentials (e.g., username and password) to the server. Assuming that this authentication succeeds, an SSH connection is said to be established between the hosts, ready for use.

The current version of the SSH protocol is 2.0. Earlier versions of the protocol have serious known vulnerabilities that preclude their use. For more information on the formal definition of the SSH protocol version 2.0, see [RFC4251], [RFC4252], [RFC4253], and [RFC4254]. All references to the SSH protocol in this publication are to version 2.0 unless explicitly stated otherwise.

3.2 Common SSH Use Cases

There are three common use cases for SSH:

- **Interactive use.** SSH is used by system administrators for manually managing and configuring Unix and Linux computers, networking equipment, and various other types of hosts remotely. SSH is also used for running applications remotely (particularly text-based legacy applications).
- **File transfers.** SSH is used as the foundation of the Secure Copy (scp) and Secure File Transfer Protocol (SFTP) protocols. These protocols are used to transfer files between hosts while leveraging the security capabilities built into SSH.
- **Point-to-point tunneling.** SSH can be used to implement a virtual private network (VPN) tunnel to protect data transmitted between two hosts. One or both of these hosts may be acting as a gateway for other hosts behind it.

[1] The primary purpose of authenticating the server is to prevent man-in-the-middle attacks.

This publication covers all of these use cases in the context of automated access. Automated access refers to accessing a host from another host in an automated fashion (without human intervention). SSH is frequently used for automated access for a variety of purposes, including managing large IT environments, integrating applications, and provisioning virtual machines in cloud services.

Automated access is commonly used with functional accounts, system accounts, service accounts, and other non-interactive user accounts (sometimes also called non-user accounts). Such accounts are used by operating systems, server applications (e.g., databases), and other applications for running processes. Automated access is also frequently used for file transfer functions.

Automated access may be unrestricted, allowing any commands to be executed, or may be limited to specific commands or operations, such as file transfers (perhaps limited to a specific directory). Organizations should limit automated access so that only the necessary commands can be executed and only the necessary resources can be engaged.

3.3 User Authentication

The SSH protocol supports several mechanisms for authenticating users, including passwords, host-based authentication, Kerberos, and public key authentication. All these authentication methods fundamentally rely on some secret information, and when used for automated access, this secret information must be stored locally or be otherwise accessible. This section briefly discusses these forms of user authentication, focusing on their relevancy and appropriateness for automated access.

3.3.1 Password Authentication

There are two kinds of password authentication mechanisms in SSH: basic password authentication and keyboard-interactive authentication. Basic password authentication is a legacy method mandated by the SSH protocol standards. Keyboard-interactive authentication is used in most modern environments, and can support challenge-response authentication and one-time passwords in addition to traditional password authentication. Password authentication is commonly used for interactive users, but less commonly for automated access, although it is sometimes seen with hard-coded passwords in scripts and management systems.

Password authentication should generally not be used for automated access because hard-coded passwords may be obtained by attackers. If password authentication is used for automated access, the passwords should be rotated frequently in accordance with the organization's password policy (which should also contain requirements such as minimum password length, minimum password complexity, etc.)

3.3.2 Host-Based Authentication

Host-based authentication uses the server host's *host key*—the key used by the client to verify the server's identity—to authenticate the source host and to vouch for the identity of the user on the client side. A configuration file (.shosts) can be used with any user account on the server to specify which users on which hosts can log into that account without further authentication. However, host-based authentication does not permit configuring command restrictions—limits on what can be done on the server with the access. Because of this, its use for automated access is not recommended.

3.3.3 Kerberos Authentication

Many organizations use Kerberos or Active Directory authentication with SSH. Kerberos (usually together with LDAP, such as in Active Directory) implements single sign-on within a Windows domain or Kerberos realm, and allows user accounts to be stored in a centralized directory. In practice, Kerberos is rarely used for non-interactive accounts. While it can be configured to use keytab files or cached tickets for functional accounts for initial authentication, these approaches rely on having long-term credentials stored on the host or at least accessible to the process on the host that is obtaining tickets. These credentials can be exploited by an attacker to obtain a ticket granting ticket (TGT) for the functional account, and the attacker can then use single sign-on or other configured Kerberos-based trust relationships to gain access to other hosts or accounts that the functional account can access.

Certain widely used SSH implementations provide single sign-on within an Active Directory domain or Kerberos realm automatically by default. Such single sign-on implies that once access has been gained to one account using Kerberos, it is possible to log in to any other server that has the same account and is in the same domain (with single sign-on permitted) without further authentication. This can easily create lots of unwanted implicit trust relationships. Another concern is that currently widely used SSH implementations do not support command restrictions for Kerberos.

Because of these problems, the use of Kerberos authentication for automated access is not recommended.

3.3.4 Public Key Authentication

Public key authentication in SSH uses user keys or certificates to authenticate a connection. Such keys can be configured for both interactive users and processes, and they authorize the user or process to access a user account in an information system. An SSH client has a user key called an *identity key*, typically an RSA or DSA private key, and the server must have the corresponding public key configured as an *authorized key* for a user account. Any user in possession of the identity key is then allowed to log into the server to that user account and perform actions under the privileges configured for the key.

For interactive users, the identity key is usually stored on a smartcard or in a passphrase-protected file in a file system on a client device. If the identity key is protected by a passphrase, it is encrypted by a key derived from the passphrase. When SSH user keys are used for automated access, however, the identity key is usually stored as an unencrypted file (with no passphrase) in the file system[2]. Given that the files grant access to servers, they contain sensitive data.

Many SSH implementations support configuring restrictions for authorized keys. These may be used for limiting what can be done on the server using the key (command restrictions) and for limiting the IP addresses from which the key can be used (source restrictions). Another advantage of public key authentication is that it does not create any implicit trust relationships, only expressly-defined trust relationships, and the permitted access can be reliably determined by inspecting the destination host.[3] This is very important for being able to audit who can access what system and account; implicit trust relationships—access from server to server that is not explicitly configured—are hard to audit and easily

[2] There is no one present to supply the passphrase for decrypting the identity key for automated access by processes. Hard-coding the passphrase in scripts would provide little additional security (see also NIST SP 800-53 IA-5(7)), and obtaining it from a vault in a script would also provide limited additional security and would be quite a maintenance burden.

[3] An exception is OpenSSH's proprietary certificate authentication, which does not allow reliably auditing who can access a host by inspecting just that host. There is no hardened Certificate Authority solution for OpenSSH, and most environments have no systematic tracking or directory of issued OpenSSH certificates. Thus if the host trusts an OpenSSH certificate signing key, it is often not possible to reliably determine who is able to access the host.

get overlooked in security policy and audits. For these reasons, public key authentication is the recommended authentication mechanism for automated access with SSH.

Public key authentication is by far the most frequently used method of configuring automated access using SSH as of this writing.

3.3.5 User Authentication Summary

Table 1 summarizes the major security problems with the user authentication methods described in this section in commonly used SSH implementations. Note that the recommended method, public key, is the only one that can avoid all of the major security problems listed in the table.

Table 1: Major Security Problems in User Authentication Methods

Authentication Form	Uses Credentials Stored on Host (e.g., hard-coded passwords) Accessible to Client User	Cannot Be Command Restricted Using Common SSH Implementations	Provides Unwanted Implicit Trust Relationships
Password	Yes	Yes	No
Host-Based	No	Yes	No
Kerberos	Yes	Yes	Yes
Public Key (Recommended method)	No[4]	No	No

[4] It depends on the configuration since the passphrase keys cannot be easily obtained from host but plaintext ones could be accessible.

4. Vulnerabilities in SSH-Based Automated Access

Management of automated access requires proper provisioning, termination, and monitoring processes, just as interactive access by normal users does. However, the security of SSH-based automated access has been largely ignored to date. Many organizations don't even know how many keys they have configured to grant access to their information systems or who has copies of those keys. These keys often grant far more access than is actually needed, such as allowing execution of any command or transfer of files to any directory. Also, in many organizations, system administrators configure new keys without any approvals or coordination, and may use them to circumvent auditing of privileged access and maintenance. Some large enterprises have hundreds of thousands or even millions of SSH user keys on their systems for automated access, which often provide many more entry points onto servers than the interactive user accounts do. Also, a sizable percentage of these keys typically grant access to administrative/root accounts or sensitive accounts, such as those storing database files or critical software.

A closely related issue is the trust relationships that these keys establish within and between systems, even between organizations. Some of these trust relationships may be undesirable or violate policies, such as leading from development and test systems into production systems, or crossing from a low-impact system to a high-impact system without requiring any additional authentication.

Although the security implications of poor SSH key management have been known for some time, the scope of the problem for automated access has not been widely understood or acknowledged. For example, most organizations do not have SSH key management or SSH-based automated access as part of their assessment programs, even though for many organizations SSH-based automated access has already become central to their identity and access management operations.

This section describes the primary categories of vulnerabilities in SSH-based automated access, and recommends possible mitigations for each category of vulnerability. The guidelines presented in subsequent sections of this document are intended to address these vulnerabilities while minimizing the administrative burden associated with the mitigations. The categories of vulnerabilities described in this section are as follows:

- Vulnerable SSH implementation
- Stolen, leaked, and unterminated SSH user keys
- Backdoors (unaudited user keys)
- Unintended usage of user keys
- Incorrect user key location

4.1 Vulnerable SSH Implementation

An SSH server or client implementation could have vulnerabilities that allow it to be exploited in order to gain unauthorized access to communications or systems. These vulnerabilities could be any of the following types:

- Software flaws in the SSH implementation (i.e., coding errors)
- Configuration weaknesses (for example, allowing the use of weak encryption algorithms)
- Protocol weaknesses (for example, supporting the use of SSH version 1)

The primary recommended mitigations for vulnerable SSH implementations are to keep all SSH server and client implementations fully up to date and to configure all SSH clients and servers securely, including preventing any use of the SSH version 1 protocol. Unfortunately, it is not always possible to upgrade, reconfigure, or otherwise alter SSH implementations (e.g., on appliances and embedded devices). In these cases, compensating controls can be used to protect the vulnerabilities, such as implementing Virtual Private Network (VPN) tunnels to encapsulate the potentially vulnerable SSH traffic.

4.2 Stolen, Leaked, and Unterminated Keys

Stolen, leaked, and unterminated identity keys pose a similar problem to stolen, leaked, and unterminated interactive user account credentials. Anyone who may have obtained a copy of an identity key—by copying it from a host, by accessing a backup, by having malware harvest keys, etc.—may use that key to attempt to gain unauthorized access to a user account on one or more servers in the organization. Once access to a user account has been gained, it is generally possible to access and modify any data for that user account—including reading and modifying the memory of processes running under that user account, and modifying any executable programs owned by that user account.

Malware can be engineered to use SSH keys to spread when automated access is allowed. The mesh of automated access relationships is so dense in many cases that it is likely that an attack can spread to most servers in an organization after penetrating the first few servers, especially if other attack vectors are used to escalate privileges.

One of the primary mitigations for stolen, leaked, and unterminated keys is to rotate SSH keys regularly, similar to how other authenticators (e.g., passwords) are changed.[5] This reduces the window of opportunity for using an SSH key and ensures that keys cannot be used indefinitely. Implicit in having SSH key rotation practices is keeping inventory of all enabled SSH identity keys in the organization, which is something that most organizations currently do not do. Note that it is generally considered infeasible to maintain a 100% inventory of all SSH identity keys because they are such small files that can be stored anywhere, even on paper (they are small enough to be typed in). The real goal is to keep an inventory of all authorized keys and to the extent possible, the corresponding identity keys.

Another primary mitigation for stolen, leaked, and unterminated keys is to implement a termination process for identity keys when an employee leaves the organization or changes roles in such a way that identity key access is no longer needed. Again, this is similar to the processes already in place for interactive user accounts and other authenticators. If a user had access to a shared identity key, that key could have been copied by the user to support continued use, so it should be rotated as soon as possible to prevent such unauthorized usage. As of this writing, most organizations do not know what each identity key is used for, so they seldom remove or rotate keys because something could break if they accidentally remove a key that is needed.

It is also fundamentally important to protect identity keys so that they are not stolen in the first place. If identity keys are only stored in the designated protected locations, then only authorized users can access them unless their hosts are compromised. A related mitigation is to only grant identity keys the minimum privileges necessary, so that if they are misused, damage will be limited. This includes minimizing the number of keys that provide administrator-level access or have privilege escalation capabilities (e.g., “sudo”).

[5] X.509v3 compliant host certificates are a useful alternative for host keys in large environments and make host key rotation much easier.

Finally, there can be limits enforced on where authorized keys can be used from and what they can be used to do, so as to make it more difficult to take advantage of stolen, leaked, and unterminated keys. Authorized keys can have SSH source restrictions configured, limiting which client computers or networks may use the keys. Alternately, firewalls and other network filtering security controls can be used to restrict SSH access to hosts with authorized keys. Forced command restrictions can be helpful in minimizing the actions that can be performed with stolen, leaked, and unterminated keys.

4.3 Backdoor Keys

Many organizations mandate that all privileged access to their servers take place through a privileged access management system that records all actions performed. Unfortunately, SSH-based automated access can be used to create a “backdoor” that bypasses the privileged access management system. This is done as easily as generating a new key pair and adding a new authorized key to an authorized keys file; authorized keys files are often not audited, so an added key may remain unnoticed for years. The corresponding identity key can then be used to log into the server using any SSH client without the privileged access management system recording the ensuing activity.

One mitigation for this threat is to ensure that only authorized administrators can modify authorized keys files. Imagine what malware could do if non-administrator users could arbitrarily grant automated access. To prevent this, authorized keys files can be stored in root-owned directories that are not writable by normal users.

Another mitigation for this threat is to monitor and audit all changes to authorized keys files. Each key addition should be supported by a documented valid purpose, and any keys added without justification should be removed immediately.

A further mitigation is to use network-level controls to force all SSH connections to go through a jump box—a type of remote access server. However, this is not realistic in many environments and may create a single point of failure for the environment. Furthermore, traditional client-based or web interface-based privileged access management systems may require extensive re-scripting to adapt to automated access. Transparent privileged access management systems, however, can audit and control both interactive and automated access without re-scripting (see Appendix C).

4.4 Unintended Usage

Users may, intentionally or unintentionally, use identity keys in ways that they were not intended to be. An example is using an identity key that was only intended to be used for automated file transfers to tunnel traffic (thus concealing it from network security controls).

One mitigation for unintended usage is to configure forced commands for authorized keys, particularly those used by external parties. This helps ensure that the intended access (such as file transfers) cannot be used for other purposes, such as executing shell commands on the server. However, it is important to know that some SSH implementations permit port forwarding even when forced commands are used; such implementations could allow an authorized key that is intended only for file transfer to be used for obtaining connections instead.

Another mitigation is to audit SSH connections, for example at a firewall, to ensure that opened SSH ports are not used for unintended purposes.

4.5 Human Errors in Key Provisioning

An authorized key may inadvertently be deployed incorrectly on a host, such as to a root account instead of a regular user account, thus granting unnecessary privileges. People are known to make human errors when manually setting up new trust relationships. Such errors can go undetected for years. Also, some key setups involve thousands of hosts, and while it is easy to miss one or more hosts when copying an authorized key to so many hosts manually, debugging such errors can be very time consuming. Also, when manually fixing problems, administrators are likely to just copy the missing keys to the proper accounts without, for example, checking whether they have accidentally been copied to the root account.

This threat can best be mitigated by automating key provisioning processes so that authorized keys are implemented exactly as they should be.

Other mitigations may reduce the likelihood that keys in erroneous locations will work properly. One example is configuring source and command restrictions for authorized keys, which limits where the keys can be used from and what they can be used to do. Another example is enforcing policies for preventing trust relationships between systems that cross security zone boundaries—for example, not permitting a low-security end user workstation to initiate an SSH connection to a high-security server.

4.6 Summary of Mitigations

- Keep all SSH server and client implementations fully up to date.
- Configure all SSH clients and servers securely, including preventing any use of the SSH version 1 protocol.
- Rotate SSH keys regularly, similar to how other authenticators (e.g., passwords) are changed.
- Keep an inventory of all enabled SSH identity keys in the organization.
- Implement a termination process for identity keys when an employee leaves the organization or changes roles in such a way that identity key access is no longer needed.
- Have a distinct, unique host key for each host using SSH for automated access.
- Protect identity keys so that they are not stolen.
- Only grant identity keys the minimum privileges necessary. This includes minimizing the number of keys that provide administrator-level access or have privilege escalation capabilities.
- Enforce limits on where authorized keys can be used from and what they can be used to do (source and command restrictions).
- Allow only authorized administrators to modify authorized keys files.
- Monitor and audit all changes to authorized keys files.
- Configure forced commands for authorized keys, particularly those used by external parties.
- Audit SSH connections to ensure that SSH is not used for unintended purposes.

- Automate key provisioning processes so that authorized keys are implemented exactly as they should be.
- Enforce policies for preventing trust relationships between systems that cross security zone boundaries.

5. Recommended Practices for Management

Managing automated access consists of implementing management, operational, and technical security control processes to address new SSH user keys, and sorting out the existing user keys to identify key replacement needs, privilege issues, etc. The objective of management is, as economically as possible, to improve security by following the NIST SP 800-53 recommended practices for access control with respect to automated access using SSH.

A project for bringing automated access using SSH user keys under management consists of the following main components:

- Establish controlled provisioning, life cycle, and termination processes for SSH user keys
- Establish continuous monitoring and audit processes to detect configured authorized keys (public keys that grant access using the corresponding private key) that have not been properly approved
- Ensure proper configuration of SSH clients and servers
- Understand and remediate existing SSH user keys and trust relationships.

Additionally, operational processes can be optimized by automating SSH user key setups and removals and related approval, documentation, monitoring, and audit processes. In particular, it is possible to integrate a key management system with a ticketing or change tracking system. If requests for provisioning automated access are made using a predefined template, a key management system can automatically do the provisioning once the request has been approved, thus reducing labor. Certain provisioning requests may touch thousands of hosts, and some organizations are known to use person-years annually on manual provisioning.

An SSH access management project may also include either selection and acquisition of software tools to aid in the process, or development of automatically or manually executed scripts.

5.1 Establish Controlled Provisioning, Life Cycle, and Termination Processes

Provisioning and configuring authenticators for automated access to an account should be a judged decision, balancing the need for access against the risks, and should include consideration of the level of access required. It is not acceptable for any user or system administrator to grant user account access to other users or processes without proper approval.

Access using authenticators intended for automated access should be properly terminated when a person leaves the organization or changes roles. Users should not be able to extend their access beyond normal termination by configuring SSH user keys for themselves or for other accounts, and should not be able to propagate access rights by configuring SSH user keys for others.

All authenticators intended for automated access, including SSH user keys, should be changed at regular intervals. They should also be changed when a compromise is suspected.

Furthermore, users should be prevented from configuring additional authenticators (e.g., SSH user keys) for accounts they access. This is necessary for the following:

- Maintaining control of who can access what information

- Properly identifying users accessing the system
- Preventing a user from accessing the system after his/her account is terminated
- Ensuring that users cannot bypass privileged access systems
- Controlling remote access
- Enforcing authorization boundaries.

A controlled process with proper documentation of approvals is necessary for being able to detect access grants by unauthorized users (e.g., unapproved authorized keys) during continuous monitoring. Without documented approvals, it will not be possible to audit that all existing authorized keys have been approved. Documenting why each key is authorized is also important for being able to remove keys when no longer needed to properly terminate access. A process should also be established for periodically revalidating the need for access and expiring keys that are no longer needed.

It is strongly recommended that authorized keys be moved to protected locations where ordinary users cannot add new keys. A technical prerequisite for establishing a controlled provisioning process is moving authorized keys to protected locations, and configuring SSH servers to only look for keys from those locations.

As part of the provisioning process, information about approved access authorizations should be stored so that it can be easily accessed, as it is needed during audits and continuous monitoring for checking that authorized keys and trust relationships have been approved and are properly configured.

In a controlled provisioning process, provisioning automated access requires:

- Submitting a request for establishing automated access to one or more accounts on one or more servers together with justification for the access. The request should specify the command that is authorized to be executed (or in exceptional cases, interactive command line access). Furthermore, a request should identify the business process, application, or information system that the request relates to and the owner/responsible person for the trust relationship if not the owner of the business process, application, or information system. The request should also specify the source account(s) and host(s) from which access is allowed or the key to be used for access. Ideally, an existing change control system is used for submitting a filled request template.
- Reviewing and approving the request based on consideration of security impact, destination account and server, authorization boundary crossing, and remote access considerations. Ideally, the change control system is used for approving the change in large enterprises or other organizations that need centralized, automated mechanisms for managing keys. The request should, at a minimum, be reviewed and approved by the official responsible for the security of the information system it grants access to. Access grants create connections between systems, and may need to be documented in security policies, plans, or architecture documents as system interconnections.
- Implementing the request, creating a key pair if needed, and configuring the authorized key(s) and identity key(s). Ideally, the implementation takes place automatically to reduce resources, reduces the need for privileged access, and eliminate manual configuration errors.

- Recording approvals for access in a database for reference (they are needed for continuous monitoring and audits). Approvals should also be periodically reviewed to determine whether the access granted by the approval is still needed.

An organization should define policy on how often approvals need to be revalidated for each information system, taking into account the impact level of the system and the level of access granted (e.g., whether the access is to a privileged account, whether execution of arbitrary commands is permitted). Ideally, in a centrally managed system the revalidation process should be continuously monitored and reflect the immediate need for individuals and systems to have access to the required resources.

5.2 Establish Continuous Monitoring and Audit Processes

The purpose of continuous monitoring is to ensure that the processes for provisioning, life cycle management, and termination are followed and enforced. It is also important for detecting unauthorized access that may have been illicitly provisioned by attackers as a backdoor into systems. Unauthorized and misconfigured SSH user keys should be detected. For example, anyone or anything (including malware) with access to a superuser account can technically configure new authenticators granting access to any account.

One approach is to periodically discover all configured authorized keys for all user accounts on all servers, check for each authorized key whether there is a corresponding valid approval (and that the key has the same access restrictions, such as command restriction, as in the approval), and generate alerts whenever the configured access does not have a matching approval. In other words, this involves establishing a baseline and whitelisting that baseline to detect any deviations from the accepted baseline. Ideally, SSH client configurations are also analyzed, and any configured identity keys are checked against approvals to ensure that the private key corresponding to each approved authorization is not found from any unauthorized locations.

Log data generated by an information system should be analyzed to 1) detect configured SSH user keys that are not being used and propose them for removal to eliminate unnecessary access grants (many of these keys likely belong to users who have left the organization); 2) detect and monitor connections from remote locations and connections crossing authorization boundaries and ensure that they are properly approved; 3) detect connections using an authorized key from hosts from which connections have not been authorized (such connections would typically involve leaked credentials and warrant immediate key rotation); and 4) identify keys that are used from outside the managed environment as “external keys” requiring manual rotation. Ideally, such log data analysis integrates with an existing SIEM or other log data collection system used by the organization.

Furthermore, continuous monitoring should ensure that remote access and non-local maintenance (including privileged access) using SSH user keys is properly audited, and that SSH/SFTP file transfers using SSH user keys and crossing a security zone boundary are properly audited and content-checked and are not used for unintended purposes (such as executing commands or transferring files in the wrong direction).

Auditing of SSH user keys serves risk analysis and ensures that the provisioning, life cycle management, and termination processes as well as continuous monitoring are working properly. A comprehensive audit of SSH user keys for risk analysis purposes should be performed by all organizations that use SSH or SFTP protocols (including within file transfer products or for network device or hardware/BIOS-level configuration). Audits for ensuring the functioning of processes, on the other hand, can use representative sampling and condition detection tests to gain sufficient confidence that the processes are functioning.

5.3 Ensure Proper Configuration of SSH Clients and Servers

SSH servers should be properly configured to establish controlled provisioning and continuous monitoring processes. Furthermore, SSH user keys are themselves security-sensitive configuration information for SSH clients and servers, and their misconfiguration, improper disclosure, or modification may expose servers to unintended access or vulnerabilities.

First, all SSH servers should be configured to log key fingerprints for access based on SSH user keys. This is necessary for performing the log analysis needed for continuous monitoring.

Second, the configuration should ensure that non-superusers cannot install new authorized keys for user accounts they use. Such new authorized keys can create backdoors, bypass privileged access auditing systems, grant permanent access, or grant access to others, without regards to controls on non-local access and authorization boundaries. In practice this means "locking down keys", i.e., moving authorized keys to superuser-owned locations that are not writable to non-superusers.

Third, automated access should be configured with the least privileges required for the intended purpose. This particularly applies to access between information systems and remote access. In practice, authorized keys should be configured with command restrictions whenever possible, and may further be configured with source restrictions, i.e., restrictions on the client IP addresses from which the keys can be used. Command restrictions are particularly important for keys used for authorizing remote file transfers to avoid accidentally permitting remote terminal access and remote execution of commands.

5.4 Understand and Remediate Existing SSH User Keys and Trust Relationships

In addition to beginning to control and manage new SSH user key setups, the existing base of SSH user keys must be inventoried, analyzed, assigned owners, and cleaned up. This process is called discovery and remediation. There are both security and compliance reasons for doing a remediation project. Existing legacy keys pose a substantial security risk and make risk analysis difficult if they are not understood.

Remediation is also necessary for compliance. NIST SP 800-53 requires identifying authorized users and their access rights, understanding and properly authorizing connections between information systems, applying the principle of least privilege, and managing authenticators, among other requirements outlined above.

A remediation project typically consists of:

- Establishing an SSH user key provisioning process
- Configuring SSH servers to log fingerprints, where needed
- Onboarding all relevant hosts into a system for managing SSH keys
- Locking down keys by moving them to root-owned locations (this is needed to produce a stable state for the later steps)
- Discovering existing configured SSH user keys on the relevant hosts

- Assigning owners (e.g., application, business process owner) for all keys granting access to servers (e.g., based on the server and account they grant access to, possibly automatically using information from a suitable configuration database)
- Monitoring log data for several months to determine which keys are not being used, proposing such keys for removal to their owners, and removing any unused keys that are not specifically approved for keeping (the monitoring period should usually be 6 to 12 months to ensure all active keys are used; the monitoring period should include testing of disaster recovery systems and other processes that may utilize keys)
- Analyzing which keys grant access across authorization boundaries or other defined boundaries (e.g., development/test to production systems, between information systems, remote access) and either categorically deconfiguring keys granting such access or obtaining required approvals for such access
- Obtaining approvals for existing authorized keys from their owners (including description of the purpose of the key or other justification for the existence of the key), and removing any keys for which such approval is not deemed appropriate (such keys may be backdoors and may have been left behind by attackers)
- Analyzing which keys are external keys, i.e., used with hosts outside the managed environment
- Rotating (i.e., changing) all identity keys and corresponding public keys (external keys may need manual handling and coordination with administrators of the external hosts with which they are used)
- Adding command restrictions to remaining keys whenever possible.

Some organizations may also wish to configure source restrictions for some of the remaining keys. Source restrictions can limit the hosts from which the key can be used, and can reduce risk from unauthorized disclosure of a key; however, they can be labor-intensive to configure and maintain as the information system evolves. An organization may wish to use them on higher-impact systems.

During the remediation project, relevant SSH configurations should be backed up before making a change so that the change can be quickly rolled back in case something breaks.

A key remediation project can be a sizable effort, but is critical for security and compliance.

5.5 Optimize the Provisioning and Termination Processes

System administrators in many organizations spend a substantial amount of time configuring and managing SSH user keys. Many organizations already use a ticketing or change control system for approving changes to their IT systems. Provisioning automated access using SSH is most naturally performed using the same system used for approving other IT changes or other access. Ideally, a special template is used for filling a request for automated access, and once approved, the request is automatically picked up by a key management system and implemented on all affected hosts. Such automation eliminates the manual steps for setting up keys, eliminates the need for manual root access for installing the keys, reduces the amount of privileged administrative access that needs to be audited and reviewed, eliminates configuration errors due to incorrectly implemented requests, and ensures that the approval template remains available for future reference and for use in continuous monitoring and audits.

6. SSH-Based Automated Access Management Planning and Implementation

This section discusses considerations for planning and implementing the management of SSH access tokens for automated access. It assumes that the organization's processes involving SSH-based automated access management are primarily ad hoc—lack of policies and requirements, lack of standardized processes and automated tools for key management, etc. As with any technology deployment, SSH-based automated access management planning and implementation should be addressed in a phased approach. A successful deployment can be achieved by following a clear, step-by-step planning and implementation process. The use of a phased approach for deployment can minimize unforeseen issues and identify potential pitfalls early in the process. This model also allows for incorporating advances in new technology and adapting the technology to the ever-changing enterprise. The following is an example of planning and implementation phases:

1. **Identify Needs.** The first phase involves identifying the needs to manage SSH access tokens for automated access and determining how those needs can best be met.

2. **Design the Solution.** The second phase involves all facets of designing the solution. Examples include cryptographic settings, key management, and management automation.

3. **Implement and Test a Prototype.** The next phase involves implementing and testing a prototype of the designed management solution in a lab or test environment. The primary goals of the testing are to evaluate the functionality, performance, scalability, and security of the management solution, and to identify any issues with the components, such as interoperability issues.

4. **Deploy the Solution.** Once the testing is completed and all issues are resolved, the next phase includes the gradual deployment of the management solution throughout the enterprise.

5. **Manage the Solution.** After the management solution has been deployed, it is managed throughout its lifecycle. Management includes maintenance of its components and support for operational issues. The lifecycle process is repeated when enhancements or significant changes need to be incorporated into the solution.

This document does not describe the planning and implementation process in depth because the same basic steps are performed for any security management technology. This section only highlights those considerations that are particular to managing SSH access tokens for automated access and were not already discussed in previous sections of this document.

Organizations may follow a project management methodology or life cycle model that does not directly map to the phases in the model presented here, but the types of tasks in the methodology and their sequencing are probably similar.

6.1 Identify Needs

The purpose of this phase is to identify the needs to manage SSH-based automated access and determine how those needs can best be met through use of automated enterprise management tools, ad hoc implementation of a key management system, etc. Requirements specific to identifying needs that should be considered include the following:

- **Existing Requirements.** There may already be mandates, regulations, organization policies, etc. that prescribe requirements for managing SSH keys used for automated access.

- **Volume of Activity.** This should include the approximate number of devices and users/processes currently using SSH-based automated access, and the approximate number of additional devices and users/processes that could benefit from SSH-based automated access.

- **System and Network Environments.** It is important to understand the characteristics of the organization's system and network environments so that a management solution can be selected that will be compatible with them. Aspects to consider include the following:

  – The characteristics of the devices that act as servers, clients, and management administrators, especially the OSs and SSH application versions (servers or clients) they use and their security postures (low-impact vs. moderate-impact, etc.)

  – The technical attributes of the interfaces of other systems with which the SSH-based automated access solution might be integrated, such as centralized logging servers and security information and event management (SIEM) software

The outcome of the analysis is the documentation of the requirements for the SSH-based automated access management solutions, including security capabilities (e.g., cryptography, key management), performance requirements, management requirements (including reliability, interoperability, scalability), the security of the technology itself, usability (by both administrators and users), and maintenance requirements (such as applying updates). These requirements will be used to design and test the solution.

6.2 Design the Solution

Once the needs and requirements have been identified, the next phase is to design an SSH-based automated access management solution that meets the needs and requirements. If these design decisions are incorrect, then the automated access implementation will be more susceptible to compromise. Major aspects of solution design that are particularly important for SSH-based automated access management and have not already been covered in this publication are as follows:

- **Cryptographic settings.** Encryption and integrity protection algorithms must be selected, as well as the key strength for algorithms that support multiple key lengths. Setting the cryptography policy involves choosing encryption and integrity protection algorithms and key lengths.[6] Federal agencies must use FIPS-approved algorithms contained in validated cryptographic modules.[7] Whenever possible, AES[8] should be used for the encryption algorithm because of its strength and speed. Several FIPS-approved algorithms are available for integrity checking, including HMAC-SHA, Cipher-Based Message Authentication Code (CMAC), and Counter with Cipher Block Chaining-Message Authentication Code (CCM).[9] Organizations should consider how easily the solution can be updated when stronger algorithms and key sizes become available in the future.

  In terms of preventing configuration weaknesses, a reasonable choice of cryptographic algorithms as

[6] NIST SP 800-21, Second Edition, *Guideline for Implementing Cryptography in the Federal Government*, presents guidelines for selecting, specifying, employing, and evaluating cryptographic protection mechanisms in Federal information systems. It defines a process for selecting cryptographic products and discusses implementation issues, including solution management, key management, and authentication. NIST SP 800-21 is available at http://csrc.nist.gov/publications/nistpubs/.

[7] The Cryptographic Module Validation Program (CMVP) at NIST coordinates FIPS 140-2 testing; the CMVP Web site is located at http://csrc.nist.gov/cryptval/. See http://csrc.nist.gov/cryptval/des.htm for information on FIPS-approved symmetric key algorithms, and http://csrc.nist.gov/cryptval/dss.htm for information on digital signature algorithms. FIPS 140-2, *Security Requirements for Cryptographic Modules*, is available at http://csrc.nist.gov/publications/fips/.

[8] For more information, read FIPS 197, *Advanced Encryption Standard (AES)*, at http://csrc.nist.gov/publications/fips/.

[9] Additional information on these algorithms is available at http://csrc.nist.gov/CryptoToolkit/modes/.

of this writing is to use AES for encryption, SHA1 (or preferably SHA2) for hashing, and HMAC-SHA1 (or preferably HMAC-SHA2-256) for Message Authentication Code (MAC). At least 2048 bit (preferably 4096 bit) DSA or RSA host keys should be used; alternatively ECDSA host keys with at least 384 bits can be used (521 bits are recommended). The diffie-hellman-group2-sha1 key exchange should NOT be used (it uses only 1024 bit Diffie-Hellman); the diffie-hellman-group14-sha1 key exchange is recommended instead (it uses 2048 bit Diffie-Hellman).

Host keys for SSH also need to be properly managed or otherwise the cryptographic security of the protocol will be compromised (hosts keys protect against man-in-the-middle attacks). Each host should generally have a distinct, unique host key, and host keys should be periodically rotated. Host certificates are a useful alternative for host keys in large environments and make host key rotation much easier.

- **Cryptographic key management and protection.** Key management and protection is another important component of solution design. Organizations should perform extensive planning of key management processes, procedures, and technologies before implementing SSH-based automated access management. This planning should include all aspects of key management, including key generation, use, storage, recovery, and destruction.[10] Organizations also need to ensure that access to keys is properly restricted.

- **Management automation.** As much of the management as possible should be automated, so as to minimize human error and resource expenditures. This includes provisioning and terminating services, as well as monitoring and auditing capabilities. Automating these management features is likely to involve acquisition of tools and/or development of scripts. Organizations should also carefully consider the security considerations inherent in the management solution, since by definition it will operate with administrative-level privileges.

- **Other security controls.** These support and complement the SSH-based automated access management implementation. For example, organizations should have policies regarding management of SSH-based automated access technologies.

6.3 Implement and Test Prototype

After the management solution has been designed, the next step is to implement and test a prototype of the design. Ideally, implementation and testing should first be performed on lab or test devices. Only implementations in final testing should be conducted on production devices. Aspects of the solution to evaluate include the following:

- **Key Management.** Each key should be properly generated, deployed, protected, rotated, and terminated. Each key should only grant the necessary access to authorized resources for authorized users/devices.

- **Administration.** Administrators should be able to configure and manage all components of the solution effectively and securely. It is particularly important to evaluate the ease of deployment and configuration, including how easily the solution can be managed as the solution is scaled to larger deployments. Another concern is the ability of administrators to disable configuration options so that users cannot circumvent the intended security. Management concerns should include the effects of changing software settings (e.g., changing cryptographic algorithms or key sizes.)

[10] NIST SP 800-57, *Recommendation for Key Management*, provides detailed information on key management planning, algorithm selection and appropriate key sizes, cryptographic policy, and cryptographic module selection.

- **Logging and Auditing.** Logging and auditing for the management solution should function properly in accordance with the organization's policies and strategies.

- **Security of the Implementation.** The management solution itself may contain vulnerabilities and weaknesses that attackers could exploit. Organizations should perform extensive vulnerability assessments against the management solution components because of their high value. Another common security concern is the security of the cryptographic keys.

Organizations should consider implementing the components in a test environment first, instead of a production environment, to reduce the likelihood of implementation problems disrupting the production environment. When the components are being deployed into production, organizations should initially use the management solution for a small number of hosts. Many of the problems that occur are likely to occur on multiple hosts, so it is helpful to identify such problems either during the testing process or when deploying the first hosts, so that those problems can be addressed before widespread deployment. A phased deployment is also helpful in identifying potential problems with scalability.

6.4 Deploy the Solution

Once testing is complete and any issues have been resolved, the next phase of the planning and implementation model involves deploying the solution. A prudent strategy is to gradually migrate devices and users/processes to the new solution. The phased deployment provides administrators an opportunity to evaluate the impact of the solution and resolve issues prior to enterprise-wide deployment. It also provides time for the IT staff (e.g., system administrators, help desk) and users to be trained.

Most of the issues that can occur during deployment are the same types of issues that occur during any large IT deployment.

6.5 Manage the Solution

The last phase of the planning and implementation model is the longest lasting. Managing the solution involves operating the deployed solution and maintaining the policies, software, and other solution components. Examples of typical actions are as follows:

- Testing and applying patches to solution components

- Including additional types of server and client devices in the automated access management solution

- Performing key management duties (e.g., issuing new keys, revoking keys for compromised systems or departing users)

- Adapting the policies as requirements change. An example is switching to a stronger encryption algorithm or increasing the key size.

- Monitoring the automated access management components for operational and security issues

- Periodically performing testing to verify that automated access management is functioning properly

- Performing regular vulnerability assessments

- Preparing devices for retirement or disposal. Devices and media that hold private keys should be sanitized or destroyed, unless the keys have been retired/rotated.[11]

[11] For more information on media sanitization, see NIST SP 800-88 Revision 1, *Guidelines for Media Sanitization* (http://csrc.nist.gov/publications/PubsSPs.html#800-88).

7. Solution Planning and Deployment

Earlier sections have provided an introduction to authentication methods in SSH, with a view towards automated access, described threats and risks associated with poorly managed SSH keys, and provided guidelines on how SSH keys should be managed. Appendix C describes tools that can be used for managing access using SSH. This section strives to answer the question, how should one practically proceed to address SSH key management issues?

A typical SSH access management or key management project involves:

- Understanding the current situation, as most organizations have a substantial installed base of SSH keys that have not previously been managed
- Procuring and deploying tools and establishing processes for efficient, controlled provisioning of automated SSH-based access and enforcing approvals according to policy
- Remediating the existing environment, including establishing purpose for every configured authorized key, removing any authorized keys and identity keys that are not used or for which no justifiable purpose can be identified, configuring command restrictions, and rotating any remaining user keys.

An understanding of the existing environment can be obtained through audits. Tools are available that can help in the audit (see Appendix C).

Deploying the tools and establishing provisioning processes typically includes locking down keys and integrating into existing IT change control or approval systems. (The integration may be performed at a later stage, or in parallel with the remediation phase.)

In a typical remediation process, each host is taken under management using the tools, each host is monitored for a period of time (preferably several months) to identify which keys are actually used and how, unused/unneeded/rogue keys are removed, trust relationships that cross boundaries inappropriately or violate policy are removed, command restrictions (and source restrictions, if applicable) are added, and all keys are rotated. The remediation process is fairly labor-intensive. Depending on how many keys there are, how readily the owner and purpose of each key can be identified, and how much work is needed for writing change requests and waiting for maintenance windows, remediating a host may take several hours of work (tools can help a lot, but they cannot determine the purpose or owner of each key if that information hasn't been recorded anywhere).

Generally it has been found that the cost of manual labor in a key management project is often as big or even significantly bigger than the cost of the tools for managing the keys. The choice of tools has a major impact on the amount of manual labor needed.

Appendix A—NIST SP 800-53 Controls Mapping

This appendix lists the NIST SP 800-53 security controls that are most pertinent for securing SSH-based automated access management. Next to each control is an explanation of its implications particular to SSH-based automated access management.

NIST SP 800-53 Controls	SSH Implications
AC-2, ACCOUNT MANAGEMENT, control #d	SSH user keys authorize access to the information system and specify privileges for access.
AC-2, ACCOUNT MANAGEMENT, control #g	Enhanced auditing of SSH should be enabled to track the usage of keys and provide an audit trail of which source user (and client) is using keys to connect to the destination user.
AC-2, ACCOUNT MANAGEMENT, control #i	Valid authorization should be required before installing an SSH authorized key, because such keys grant access to the system. Individual users or administrators should not be able to grant access to friends or colleagues without control. Granted key-based access should be limited to intended usage (e.g., intended command).
AC-2, ACCOUNT MANAGEMENT, control #j	SSH keys should be monitored periodically for compliance with key management policies.
AC-2, ACCOUNT MANAGEMENT, control #k	Any private keys held by a group of individuals should be rotated whenever an individual is removed from the group (note: administrators may obtain copies of keys when using service accounts). Keys stored on shared accounts on jump servers should be rotated when someone's access to the jump server is terminated.
AC-3, ACCESS ENFORCEMENT	Approvals for key-based access should be enforced. Users should not be able to install unapproved keys (keys must be locked down).
AC-3, ACCESS ENFORCEMENT, control enhancement #3	Users should be prevented from propagating their access rights by installing new private keys.
AC-4, INFORMATION FLOW ENFORCEMENT	SSH keys should be managed in order to have control over the flow of information between interconnected systems.
AC-4, INFORMATION FLOW ENFORCEMENT, control enhancement #4	Content of encrypted SSH sessions and SFTP file transfers should be content-checked (e.g., run through data loss prevention (DLP) software, antivirus software).
AC-6, LEAST PRIVILEGE	Command restrictions should be configured for SSH authorized keys whenever possible to limit what can be done with the keys. Key-based access to root accounts should be limited to cases where it is necessary to accomplish the task. Only those authorized keys that are necessary for accomplishing the assigned tasks should be configured. In corollary, it is necessary to know what task each key relates to.
AC-6, LEAST PRIVILEGE, control enhancement #2	SSH authorized keys for privileged accounts should only be configured if the task cannot be accomplished using a non-privileged account.

AC-6, LEAST PRIVILEGE, control enhancement #3	SSH authorized keys providing root or other privileged level access should be approved and documented before they are provisioned. Approved authorized keys should specify the command they authorize to be executed. An inventory of approved authorized keys and trust relationships should be maintained, documenting the rationale for each authorized key.
AC-6, LEAST PRIVILEGE, control enhancement #4	Unauthorized users should not be given access to private keys that grant access to privileged accounts.
AC-6, LEAST PRIVILEGE, control enhancement #5	Organizations should ensure that non-organizational users cannot obtain copies of private keys and that any such copies are rendered inoperative by regular key rotation.
AC-6, LEAST PRIVILEGE, control enhancement #7	Privileges granted by SSH keys should be reviewed and adjusted to reflect organizational/business needs. Inappropriate, unneeded, or unused authorized keys should be removed and command and source restrictions added appropriately.
AC-6, LEAST PRIVILEGE, control enhancement #9	Encrypted SSH connections accessing privileged accounts should be audited.
AC-6, LEAST PRIVILEGE, control enhancement #10	Authorized keys files should be locked down to prevent users from adding their own or other public keys to privileged accounts without formal provisioning or approvals.
AC-17, REMOTE ACCESS	Policy should state whether to allow SSH key-based remote access, and if so, state requirements for key rotation and how to prevent copying of private keys (e.g., key must be stored on smartcard). Remote access should be expressly authorized (not provisioned by individual users), which implies locking down keys.
AC-17, REMOTE ACCESS, control enhancement #2	SSH provides such mechanisms. Host key management should be required for preventing man-in-the-middle attacks.
AC-17, REMOTE ACCESS, control enhancement #4	The purpose of each authorized key granting access to root or other privileged accounts should be documented prior to provisioning access and such should be limited to [the organization-defined needs].
AC-20, USE OF EXTERNAL INFORMATION SYSTEMS	SSH user keys can be used for external access. Terms and conditions should address 1) whether and in what direction is key-based access allowed; 2) whether command restrictions are required; 3) whether privileged external access is permitted; 4) what can be done within each external connection (e.g., is port forwarding allowed); and 5) is DLP required for content.
AU-3, CONTENT OF AUDIT RECORDS, control enhancement #1	Enhanced auditing of SSH should be enabled to track the usage of keys and provide an audit trail of which source user (and client) is using keys to connect to the destination user.
CA-2, SECURITY ASSESSMENTS	Many organizations have more SSH user key based authenticators enabling access than they have interactive accounts. Therefore, SSH user key based access should be addressed in the security assessment plan and included in the security assessment.

CA-3, SYSTEM INTERCONNECTIONS	SSH user keys define permanent trust relationships that interconnect information systems (possibly allowing anyone gaining privileged access to one information system to gain unrestricted access to the other information system). SSH key-based trust relationships between information systems must be 1) expressly authorized; 2) documented (including security requirements, such as key rotation); and 3) periodically reviewed.
CA-5, PLAN OF ACTION AND MILESTONES	Since unmanaged SSH user keys can undermine so many other security controls, the action plan should include corrective actions to address SSH user key-based security weaknesses and deficiencies noted during the assessment.
CA-7, CONTINUOUS MONITORING	SSH-based access should be regularly analyzed as part of a continuous monitoring program to detect unauthorized authorized keys configured by users or system administrators (necessary for maintaining and enforcing security authorizations in dynamic environments).
CA-10, INTERNAL SYSTEM CONNECTIONS	Intra-system SSH connections should be properly authorized and documented. Typically such connections are configured using SSH user keys, so authorization and documentation should be required before provisioning authorized keys.
CM-3, CONFIGURATION CHANGE CONTROL	SSH keys and SSH configuration files are security-sensitive configuration information, and their misconfiguration, modification, or unauthorized disclosure may expose servers to unintended access or vulnerabilities. Thus, changes to them should be reviewed, documented, and audited.
CM-5, ACCESS RESTRICTIONS FOR CHANGE	Provisioning and configuring authenticators for automated access to an account should be a controlled, judged decision, balancing the need for access against the risks and must include consideration of the level of access required. Users and system administrators should not be able to modify SSH keys and configuration files without oversight.
CM-6, CONFIGURATION SETTINGS	Standard configurations for sshd_config settings (including AuthorizedKeysFile and PermitRootLogin parameters) should be defined and applied to all SSH servers. SSH key configurations should be documented and approved. Changes should be monitored.
CP-2, CONTINGENCY PLAN	SSH user keys are frequently used in systems that copy data to disaster recovery sites and implement switching operations to use disaster recovery sites. Poorly managed SSH trust relationships (e.g., ones without command restrictions) may be used to spread an attack between sites. The contingency plan should not rely on switching operations to disaster recovery sites if a trust relationship without a command restriction permits access from one site to another, allowing attacks and malware to spread between sites.
IA-2, IDENTIFICATION AND AUTHENTICATION (ORGANIZATIONAL USERS)	SSH identity keys should be associated with an individual user. Policy should prohibit users from sharing private keys, using another user’s private key, or copying/moving a private key to another system. Monitoring of key usage should be performed to detect instances where keys are being shared by multiple users, and compromised/shared keys should be rotated.
IA-3, DEVICE IDENTIFICATION AND AUTHENTICATION	Unique SSH host keys should be used for every host running SSH.

IA-5, AUTHENTICATOR MANAGEMENT	SSH user keys (particularly identity keys) are authenticators. SSH identity key creation and corresponding authorized key installation should involve administrative procedures. Lifetime of SSH user keys should be limited. They should be periodically rotated. Policy should prohibit users from sharing private keys, using another user's private key, or copying/moving a private key to another system. Continuous monitoring and regular key rotation should be used to enforce the policy. Shared accounts on jump servers are group/role accounts, and any private keys stored on such accounts (for access from the jump server to end hosts) should be changed when a user's access to the shared account is terminated (effectively, membership in the group of people with access to the account is terminated).
IA-5, AUTHENTICATOR MANAGEMENT, control enhancement #7	Password-based automated access using passwords coded in scripts or binaries should be prohibited.
IA-8, IDENTIFICATION AND AUTHENTICATION (NON-ORGANIZATIONAL USERS)	The same SSH private key (identity key) should not be used for more than one user.
MA-2, CONTROLLED MAINTENANCE	SSH configurations and authorized keys should be checked after maintenance to ensure they are still properly configured and functioning (e.g., as part of continuous monitoring).
PL-2, SYSTEM SECURITY PLAN	Automated access should be considered when defining and enforcing the authorization boundary. A system that can execute commands on another system (e.g., using SSH keys) should be considered to have at least the same security categorization as the system(s) it can access. Trust relations and automated SSH connections with/to other systems should be documented in the security plan.
PL-8, INFORMATION SECURITY ARCHITECTURE	Automated access and SSH key-based access must be considered in the information security architecture. Given that many organizations have more SSH user keys granting access than they have PKI tokens or passwords, they cannot be ignored when developing the information security architecture.

PS-4, PERSONNEL TERMINATION	SSH keys are authenticators that may be associated with an individual. Some individuals may also have had access to authenticators used for shared accounts or privileged accounts (e.g., SSH private keys stored on servers). Any authenticators associated with an individual should be terminated upon the individual's employment - this means at minimum removing the related authorized keys from all servers. Any authenticators that the individual may have had access to should be rotated (i.e., changed) to ensure that copies of the authenticators do not remain operable.
PS-6, ACCESS AGREEMENTS	Access should not be provisioned without proper access agreement. This is one more reason why users or system administrators provisioning key-based access to themselves or others without proper approvals is not acceptable.
RA-3, RISK ASSESSMENT	Risk assessment should consider attack propagation risk: if one system is compromised, an attack could spread to other systems using SSH keys (especially if command restrictions are not used), possibly even to backup systems and disaster recovery sites. The potential impact of an SSH key compromise should be part of the risk assessment process. Authorized keys grant access to an information system. It is impossible to assess the risk facing an information system without knowing and having control of who can access the system and what other systems the system under inspection can access without further authentication.
SC-12, CRYPTOGRAPHIC KEY ESTABLISHMENT AND MANAGEMENT	SSH keys should be rotated and reissued on a regular basis. SSH user keys should be created only through a controlled provisioning process.
SC-23, SESSION AUTHENTICITY	Unique host keys should be used for every SSH server, as that prevents man-in-the-middle attacks.
SI-4, INFORMATION SYSTEM MONITORING	SSH connections using key-based authentication should be monitored and checked against authorized connections to detect unauthorized use of authorized keys (e.g., using copied identity keys) in real time.
SI-7, SOFTWARE, FIRMWARE, AND INFORMATION INTEGRITY	Authorized keys and other SSH configuration files installed without proper authorization should be detected (e.g., using continuous monitoring tools).

Appendix B—Cybersecurity Framework Subcategory Mapping

This appendix lists the Cybersecurity Framework subcategories that are most pertinent for securing SSH-based automated access management. Next to each control is an explanation of its implications particular to SSH-based automated access management.

Cybersecurity Framework Subcategory	SSH Implications
ID.AM-3: Organizational communication and data flows are mapped	SSH user keys define permanent trust relationships that interconnect information systems (possibly allowing anyone gaining privileged access to one information system to gain unrestricted access to the other information system).
ID.RA-5: Threats, vulnerabilities, likelihoods, and impacts are used to determine risk	Risk assessment should consider attack propagation risk: if one system is compromised, an attack could spread to other systems using SSH keys. The potential impact of an SSH key compromise should be part of the risk assessment process. Authorized keys grant access to an information system. It is impossible to assess the risk facing an information system without knowing and having control of who can access the system and what other systems the system under inspection can access without further authentication.
PR.AC-1: Identities and credentials are managed for authorized devices and users	SSH user keys authorize access to the information system and specify privileges for access. SSH identity keys should be associated with an individual user. Unique SSH host keys should be used for every host running SSH.
PR.AC-3: Remote access is managed	SSH can provide key-based remote access. Remote access should be expressly authorized (not provisioned by individual users). There should be requirements for remote access, such as requirements for key rotation and preventing copying of private keys.
PR.AC-4: Access permissions are managed, incorporating the principles of least privilege and separation of duties	Command restrictions should be configured for SSH authorized keys whenever possible to limit what can be done with the keys. Key-based access to root accounts should be limited to cases where it is necessary to accomplish the task. Only those authorized keys that are necessary for accomplishing the assigned tasks should be configured.
PR.DS-2: Data-in-transit is protected	SSH connections using key-based authentication protect the confidentiality and integrity of data in transit.
PR.DS-6: Integrity checking mechanisms are used to verify software, firmware, and information integrity	Authorized keys and other SSH configuration files installed without proper authorization should be detected (e.g., using continuous monitoring tools).
PR.IP-3: Configuration change control processes are in place	SSH keys and SSH configuration files are security-sensitive configuration information, and their misconfiguration, modification, or unauthorized disclosure may expose servers to unintended access or vulnerabilities. Thus, changes to them should be reviewed, documented, and audited.
PR.PT-1: Audit/log records are determined, documented, implemented, and reviewed in accordance with policy	Enhanced auditing of SSH should be enabled to track the usage of keys and provide an audit trail of which source user (and client) is using keys to connect to the destination user.

Cybersecurity Framework Subcategory	SSH Implications
PR.PT-3: Access to systems and assets is controlled, incorporating the principle of least functionality	SSH user keys define permanent trust relationships that interconnect information systems (possibly allowing anyone gaining privileged access to one information system to gain unrestricted access to the other information system). SSH key-based trust relationships between information systems must be 1) expressly authorized; 2) documented (including security requirements, such as key rotation); and 3) periodically reviewed.
DE.CM-7: Monitoring for unauthorized personnel, connections, devices, and software is performed	SSH-based access should be regularly analyzed as part of a continuous monitoring program to detect unauthorized authorized keys configured by users or system administrators (necessary for maintaining and enforcing security authorizations in dynamic environments).

Appendix C—Tool Selection

At a minimum the following areas should be considered when selecting and procuring tools for managing SSH keys:

- What kind of process does the product support for provisioning and terminating access? Can it automatically implement requests approved in a change control system? How does the process match the organization's needs?
- How does the product implement continuous monitoring, particularly access to and analysis of syslog data and other data (e.g., configuration management databases)?
- What is the deployment model, scalability, security model, and availability architecture of the product and does it meet the organization's needs?
- How well does the product support the remediation process, particularly identifying unused keys, identifying keys that cross boundaries in ways that violate policy (e.g., DEV -> PROD), detecting unauthorized changes, configuring command and source restrictions, and auditing of the state of the environment?

Given that labor costs of the remediation project and the cost of ongoing key provisioning form the bulk of total costs around managing automated SSH-based access, it is important to consider these aspects of a project and select a product that minimizes the total cost of the project - this may not necessarily be the lowest cost product.

A well-designed product can produce operational cost savings and pay for itself by reducing manual labor previously expended on SSH key provisioning, termination, and rotation - and improve operational flexibility by reducing lead times in provisioning - in addition to meeting the security requirements driving the project.

If the organization does not have sufficient understanding of SSH keys, automated access management, and the remediation process in-house, use of consultants to assist in the process may be warranted.

Providers of tools that may assist in managing and implementing automated access using SSH keys include the following:

Fox Technologies (http://www.foxt.com/)

- **FoxT ServerControl**: A product for server management, including management, inventory, and recycle SSH server keys.

SSH Communications Security, Inc. (http://www.ssh.com/)

- **Universal SSH Key Manager**: A product for managing the full SSH access management process for automated and interactive access using SSH user keys and helping the organization through the remediation process.
- **Cryptoauditor:** A product for transparent privileged access management of SSH and Remote Desktop connections.

- **SSH Risk Assessor:** A product for quantifying SSH key risks and auditing SSH-related policy violations.

Venafi, Inc. (http://www.venafi.com/)

- **Venafi Director**: A certificate and encryption key management product, including discovery of and visibility to the inventory of SSH keys in an organization.

Appendix D—Building Block

This appendix contains a detailed discussion of a particular problem that is relevant across a variety of industry sectors. The discussion essentially constitutes a "building block" for addressing the challenge.

D.1 Description

Automated access between systems has become increasingly prevalent. Examples of this access include file transfers, disaster recovery, privileged access management, software and patch management, and dynamic cloud provisioning. Automated access is often accomplished using SSH; however, these architectures have been developed on an ad hoc basis, without formal lifecycle management. This has caused a variety of security issues that may leave SSH-enabled systems more susceptible to compromise and may make these systems less well-audited, therefore reducing accountability.

A possible solution for improving the security of SSH-enabled systems that allow automated access is to implement management, operational, and technical security control processes that address the issuance and maintenance of new SSH user keys. This includes auditing the existing user keys to identify key replacement needs, privilege issues, and other problems. The primary drawback with this approach is the amount of effort that it may involve, and the resulting expenses that an organization may have to pay. Imagine having to audit hundreds of thousands of existing user keys within a large organization. So the goal is to establish an automated access management capability that improves security without overwhelming the resources of the organization.

D.2 Scenario

Example Scenario – Existing SSH User Keys

An organization has used SSH user keys to enable system-to-system access for years. There was no record keeping for these keys and no other lifecycle management activities; keys were simply issued as requested. To get the situation under control, the organization first implements new policies and procedures for handling all user keys in the future, including issuing, replacing, and revoking them. Next, the organization audits the existing server configurations and keys to determine what is needed. Because the new processes are in place, the old keys can be handled under these processes—such as issuing new keys and revoking old keys. Once all old keys have been revoked (after being replaced with new keys or determined to be unnecessary), then the new keys can all be managed under the new lifecycle management structure.

D.3 Desired Solution Characteristics

- Only the necessary SSH user keys are issued. Keys are revoked when no longer needed. Keys that are needed over a period of time are rotated on a regular basis during that time.
- Each SSH user key has only the necessary privileges/functionality.
- Detect suspicious activity more quickly so that inadvertent or intentional damage to organization data and resources can be reduced

D.4 Approach

This building block is intended to demonstrate security capabilities that can better safeguard automated access between systems. As described throughout this publication, implementing a variety of management, operational, and technical security controls is proposed as the fundamental approach to solving this problem.

One of the most important parts of this solution is the audit of existing SSH user keys. This audit is critically necessary for determining which SSH user keys are necessary and restricting each key to only the required privileges/functions. It is important to note that the audit is strictly an inventory and assessment exercise; it does not directly involve actions with keys, such as reissuing or revoking them.

Another important facet of this solution is ensuring that future activities involving SSH user keys are controlled and monitored. Formal processes, not ad hoc actions, need to be followed to ensure that only the necessary keys are issued, that the keys are secured, that the keys are only used as designated, and that keys are reissued, revoked, etc. as necessary. If these formal processes are not put into place and maintained themselves, security conditions will tend to rapidly deteriorate and the organization will soon be in a position where another full-fledged enterprise-wide user key audit will be needed.

D.5 Business Value

- Enable rapid secure deployment of automated system-to-system access for data exchange purposes
- Prevent unauthorized usage of old SSH user keys, thus preventing potential compromises of organization data and resources
- Prevents unnecessary and unauthorized data accesses/transfers, which could compromise sensitive organization data

D.6 Relevant Standards

- Ylonen, T., Kent, G., and Souppaya, M., "Managing SSH Keys for Automated Access – Current Recommended Practice", Internet-Draft, April 2013. https://tools.ietf.org/html/draft-ylonen-sshkeybcp-01

Appendix E—Acronyms and Abbreviations

Selected acronyms and abbreviations used in this publication are defined below.

BIOS	Basic Input/Output System
FIPS	Federal Information Processing Standard
FISMA	Federal Information Security Management Act
IT	Information Technology
ITL	Information Technology Laboratory
LDAP	Lightweight Directory Access Protocol
NIST	National Institute of Standards and Technology
OMB	Office of Management and Budget
OS	Operating System
PIV	Personal Identity Verification
RFC	Request for Comment
SFTP	Secure File Transfer Protocol
SP	Special Publication
SSH	Secure Shell
TGT	Ticket Granting Ticket
VPN	Virtual Private Network

Appendix F—Glossary

Selected terms used in this publication are defined below.

Authorized Key: A public key that has been configured as authorizing access to an account by anyone capable of using the corresponding private key (identity key) in the SSH protocol. An authorized key may be configured with certain restrictions, most notably a forced command and a source restriction.

Automated Access: Access to a computer without an interactive user, generally machine-to-machine access. Automated access is often triggered from scripts or schedulers, e.g., by executing an SSH client or a file transfer application. Many programs may also use automated access using SSH internally, including many privileged access management systems and systems management tools.

External Key: An authorized key that is used from outside the organization (or outside the environment considered for SSH user key management purposes), or an identity key that is used for authenticating to outside the organization (or outside the environment considered for SSH user key management purposes). Key rotation can break external keys, and therefore it must be ensured that the other side of trust relationships involving external keys is also properly updated as part of rotation. Alternatively, rotation of external keys may be prevented, but that is not a sustainable solution long-term.

Fingerprint: A hash value of a (public) key encoded into a string (e.g., into hexadecimal). Several fingerprint formats are in use by different SSH implementations.

Forced Command: A restriction configured for an authorized key that prevents executing commands other than the specified command when logging in using the key. In some SSH implementations, forced command can be configured by using a "command=" restriction in an authorized keys file.

Host Key: A public key used for authenticating a host in the SSH protocol to hosts that want to communicate with it (each host also generally has its own private host key). Some hosts may have more than one host key (e.g., one for each algorithm). Host keys are used for authenticating hosts (machines) themselves, not users or accounts, whereas identity keys and authorized keys relate to authenticating users/accounts and authorizing access to accounts on hosts.

Identity Key: A private key that is used for authentication in the SSH protocol; grants access to the accounts for which the corresponding public key has been configured as an authorized key.

Key: A cryptographic key. In this document, keys generally refer to public key cryptography key pairs used for authentication of users and/or machines (using digital signatures). Examples include identity key and authorized keys. The SSH protocol also uses host keys that are used for authenticating SSH servers to SSH clients connecting them.

Key Rotation: Changing the key, i.e., replacing it by a new key. The places that use the key or keys derived from it (e.g., authorized keys derived from an identity key, legitimate copies of the identity key, or certificates granted for a key) typically need to be correspondingly updated. With SSH user keys, it means replacing an identity key by a newly generated key and updating authorized keys correspondingly.

Source Restriction: A restriction configured for an authorized key that limits the IP addresses or host names from which login using the key may take place. In some SSH implementations, source restrictions can be configured by using a "from=" restriction in an authorized keys file.

User Key: A key that is used for granting access to a user account in the SSH protocol (as opposed to a host key, which does not grant access to anything but serves to authenticate a host). Both authorized keys and identity keys are user keys.

Appendix G—References

References for the publication are listed below.

[ID-SSH] Ylonen, T., Kent, G., and Souppaya, M., "Managing SSH Keys for Automated Access – Current Recommended Practice", Internet-Draft, April 2013.

[RFC4251] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[RFC4252] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[RFC4253] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC4254] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Connection Protocol", RFC 4254, January 2006.