What Every CISO Must Know About SSH Keys

Secure Shell (SSH), also known as Secure Socket Shell, is a cryptographic network protocol for operating network services securely over an unsecured network. SSH keys resemble passwords in that they permit privileged access to systems at the operating system level for users or other applications. SSH keys are used for automation, backups, data copying, application integration, automation and systems management. Mismanagement of these keys represents a significant security risk of which chief information security officers (CISOs) must be aware so they can be prepared to take action to prevent misuse and data losses.

> **"SSH keys grant access to systems in the same way that passwords do, but without the need for a password to be entered."**

SSH implementations are preinstalled in all Unix, Linux and Mac systems and are widely used on Windows, routers and telecommunications networks as well. Thus, SSH has become the tool of choice for automated access between systems, commonly implemented by system administrators without any higher controls or audits. Many organizations do not even have a policy regarding SSH key-based access.

SSH keys grant access to systems in the same way that passwords do, but without the need for a password to be entered. In fact, one can think of SSH keys as more elaborate passwords, implemented using public key cryptography. A public-key is configured as an authorized key on a server to which access is granted; the corresponding private key is configured as an identity key on the server from which access is granted.

Access authorization using authorized keys and identity keys uses what is called public key authentication in the SSH protocol. In essence, the client digitally signs a session-specific value using the identity key and the server uses the authorized key to verify the digital signature.

Locks and Keys

Analogous to the physical world, one can think of each user account on a server as having a lock, and the password and authorized keys identify the keys that open the lock. Usually, there is only one password per account, but there can be any number of authorized keys. Identity keys are like physical keys; just as a physical key opens a lock, an identity key enables logging into the account at the operating-system level. Management of SSH keys, then, is a control issue because the access they provide is often overlooked.

In more technical detail, on most Unix/Linux systems, the authorized keys for each account are listed in the $HOME/.ssh/authorized_keys file, and the identity keys are usually stored in a user's $HOME/.ssh directory (but can be anywhere, even offline). This means that it is possible to audit which keys are given access to each server and account, but it will never be possible to fully determine who has a copy of the private key. Many hackers say that private keys are the first thing they collect because, if the system owner does not manage them properly,

Tatu Ylonen
Is chief executive officer and board member of SSH Communications Security, and author of US National Institute of Standards and Technology Internal Report 7966, *Security of Interactive and Automated Access Management Using Secure Shell (SSH),* and several Internet Engineering Task Force standards. Ylonen is an inventor and holds more than 30 US and international patents, including several patents on major telecommunications standards. Prior to his current position, Ylonen held various roles including chief operating officer and chief technology officer.

private keys are probably the least risky way (to the hacker) of leaving a backdoor into a system.

On Windows, private-key storage is highly specific to the SSH server software being used and can be checked only by using implementation-specific tools or SSH key management tools.

The primary reason for using public-key authentication is that it allows unattended access. Nobody needs to be present to type a password. The identity key is usually stored in a file, and anyone with access to that file can log into any server with the corresponding authorized key or execute commands on that server. The most common application of this is automated file transfers. Other kinds of systems management automation are also very common. Tools such as Puppet, Chef, Ansible, Bladelogic and many file transfer solutions use SSH public key authentication internally. The SSH File Transfer Protocol (SFTP) runs over SSH.

To give an idea of the scope of SSH usage, one typical major bank that the author worked with closely has more than five million automated key-based SSH logins every day, as determined from syslog data. It would not be possible to run modern, large, highly integrated IT systems without SSH and the associated automation. That pairing is scattered throughout thousands and thousands of scripts and hundreds of application teams.

There is also another kind of SSH key: the host key. It is used for authenticating hosts rather than users, and its primary purpose is man-in-the-middle attack prevention. Without the host key (or if the host key is not properly managed), an attacker can fool a client to connect to an intermediate server and send a password to the intermediate server. The intermediate server can then steal the password and make a connection to the end server so that the user does not notice anything. Cain & Abel, jmitm2 and sshmitm are popular tools for such attacks. It is important to properly manage host keys to prevent password stealing. Public key authentication with authorized keys, however, is not vulnerable to this attack.

There is a lot of SSH key-based access inside most organizations. There is also a surprising amount of it in Windows environments. There are stunning numbers of SSH keys that grant access—there could be 10-50 times as many as there are usernames and passwords (employees) in the organization. What needs to be done to keep these keys safe?

Granting Access

Prudent information security starts with controlling who can access systems. It is necessary to know who is given legitimate access, i.e., who is managing users and credentials (whether they be passwords, certificates, smartcards or SSH keys). Enterprises should give access only when there is a legitimate need, give only the least amount of access needed to perform the task, and terminate the access when it is no longer needed, the employee leaves or the contract with a third party ends. These guidelines are common sense, and they are mandated in, for example, the US Federal Information Security Management Act (NIST SP 800-53r4 AC-2, AC-6, PS-4), Payment Card Industry Data Security Standard (Sections 7.1, 8.1, 6.4.1), the US Health Insurance Portability and Accountability Act (CFR 164.308), North American Electric Reliability Corporation Critical Infrastructure Protection (CIP-003-3 R5), and COBIT®. Credentials for root and service accounts must also be properly managed and are often more critical than normal user credentials.

Failure to control who can access systems and to properly terminate access when it is no longer needed is reckless and negligent. US law (33 USC 3552[b][3]) defines information security as "protecting information and information systems against unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide … integrity … confidentiality … availability."[1] None of these objectives is achieved if SSH key-based access is not properly controlled.

In many organizations, system administrators install new SSH keys at will whenever convenient. These

keys may be installed without approvals from the involved application teams and without any (reliable) registry of them. At the other end of the spectrum, the author has interacted with an enterprise that has a 15-person dedicated, centralized security administrator team installing SSH keys.

> **"In many organizations, system administrators install new SSH keys at will whenever convenient."**

In the author's experience, about 10 percent of configured authorized keys grant root access. The exact percentage varies from organization to organization. In almost all organizations, the total number of authorized keys is equal to five to 50 times the number of employees.

The author's analysis of how the existing SSH keys are being used (using syslog data and other information) shows that, in many cases, about 90 percent of the configured authorized keys are no longer being used at all. This may occur because the employee associated with a specific key left or perhaps because the integration need, audit or other purpose for which the key was installed has ceased to exist, but the access was never terminated.

Lack of Restriction Creates Risk

SSH keys allow restricting privileges that are given to them using command restrictions (the "command=" option in $HOME/.ssh/authorized_keys). System administrators rarely restrict what can be done on a host to any specific command, instead allowing for unrestricted jumping from one host to another, from one information system or application to another, or from one data center to another. This violates the principle of least privilege access and breaks internal boundaries, allowing attacks to spread from one server to another and from one data center to another—sometimes also to disaster recovery and backup systems.

An additional issue is that many organizations do not restrict automated access from test and development systems to production. Such DEV->PROD access is expressly forbidden by, for example, the Singapore Monetary Authority rules.[2] It is also common sense to limit this type of access; test and development systems do not have the same level of security as critical production systems, and if someone can log in from a test system to a production system without a password, the production system is no more secure than the test system. Such access also violates separation of duties.

Best Practices in Key Management

Due to the highly technical and distributed nature of SSH keys, they have been ignored by many auditors. Only in recent years have they started to get more attention and their impact properly understood. Given that the author has seen that, in many organizations,

more than 90 percent of credentials granting access to production servers are SSH keys, it is clear that they cannot continue to be ignored. Leading IT auditors are becoming more savvy with SSH keys.

The following steps can be taken to keep SSH keys secure:

1. The issue of unmanaged SSH keys must be recognized and someone must be assigned the responsibility for looking into it. Otherwise, it will fall through the cracks, resulting in no resolution and continued risk for the enterprise.
2. The organization's SSH key situation must be assessed. A proper audit can provide a thorough picture; however, at this time external auditors only sample the environment.
3. Existing keys must be taken under control. This includes:
   - Discovering all authorized keys (and identity keys to the extent possible) in the environment
   - Remediating the existing situation's vulnerabilities. This includes (among other things) identifying and removing keys that are no longer used and identifying a legitimate business need and responsible person/team for each remaining authorized key.
   - Identifying and eliminating access from test and development systems to production/disaster recovery (or, in the rare cases where it is to remain, properly justifying, understanding and documenting the access)
4. A process and policy must be established for requesting, approving, configuring and terminating new keys.
5. Eventually, all existing keys must be changed (rotated) so that who has copies of them can be known.

What Matters in Key Management

Software is almost always needed for implementing SSH key discovery, remediation and monitoring. Many organizations also want to move private keys to root-owned locations (under/etc or/var) to prevent application administrators and ordinary users from adding uncontrolled authorized keys (key lock-down).

Many people also want to add source restrictions ("from=" option in the authorized_keys file) to limit the hosts from which a key can be used. This restriction does not protect against active network-level attacks, but it can make the use of stolen identity keys more difficult.

Key sizes and algorithms are not central in this context. While security policies should state sufficient key sizes (e.g., at least 2048-bit RSA), any default key size is still probably acceptable or at least a much smaller risk than having thousands of unaudited authorized keys. Even the first SSH versions created 1024-bit RSA keys by default, and while this is no longer recommended, the key size is not really a security risk in this use case.

> **"SSH key management can best be tackled by the core security teams."**

SSH key management is also not a traditional privileged access management issue. SSH keys are primarily an automation tool and access configuration tool. Furthermore, anyone with access to an identity key can bypass the jump server of a privileged access management system and log in directly to the destination host. Putting private keys in a vault does not solve the real problem (but it does prevent having to manage them with any other tool, which would create vendor lock-in). A real solution requires addressing authorized keys.

A Coordinated Effort

SSH key management can best be tackled by the core security teams. They understand the implications of access between systems, look at the overall situation in the organization and have the required policy-setting authority. SSH key

management is an access management issue. The main challenge with it is that many access management professionals are too focused on passwords and integration to human resource systems to understand the automation aspects of it. Platform engineering teams (e.g., Unix engineering) are probably able to understand automation aspects. It is not really an encryption or public key infrastructure (PKI) issue, even though those teams would be familiar with the concept of a key pair.

In practice, an SSH key management project needs cooperation from security, access management, compliance, Unix engineering, Unix operations, Windows operations and application teams. Often, risk management or audit is also involved. Someone with sufficient authority to begin this effort—the CISO makes the most sense in this context—needs to support the project. The US National Institute of Standards and Technology Internal Report (NISTIR) 7966 is a useful resource for enterprises beginning the process of securing their SSH keys.[3]

Endnotes

1 Government Printing Office, *Federal Information Security Modernization Act of 2014*, USA, December 2014, *https://www.gpo.gov/fdsys/pkg/PLAW-113publ283/html/PLAW-113publ283.htm*

2 Monetary Authority of Singapore, *Guidelines on Risk Management Practices—Internal Controls*, March 2013, *www.mas.gov.sg/~/media/MAS/Regulations%20and%20Financial%20Stability/Regulatory%20and%20Supervisory%20Framework/Risk%20Management/RMG%20Internal%20Control_1%20Apr%202013*

3 Ylonen, T.; P. Turner; K. Scarfone; M. Souppaya; *NISIR 7966 Security of Interactive and Automated Access Management Using Secure Shell (SSH)*, October 2015, *http://nvlpubs.nist.gov/nistpubs/ir/2015/NIST.IR.7966.pdf*