Secure SSH: Risk Management Solutions

Greg Kent
Vice President, SecureIT

SECURE IT

Secure SSH: Risk Management Solutions

Introduction

This eBook discusses risk management for SSH implementations, including key discovery, establishing a key association registry, and centralizing key provisioning. It is Part 2 of a series providing a comprehensive overview of managing risks associated with SSH.

Controlling Key Associations

Unknown or inappropriate key associations can provide an access control backdoor that bypasses controls focusing on passwords. Consequently, managing key associations is an essential part of having a well controlled SSH environment. In the context of authentication controls, it is essential to be able to determine exactly which accounts on the source systems have access to connect to which accounts on the target systems. In the SSH architecture, it is the key associations, also called trust relationships, that provide this information.

The Identity File Dilemma

In the default OpenSSH implementation, getting a handle on key associations can be quite a challenge. Imagine that Information Security or Internal Audit identifies a key association with a particular public key. In order to maintain the security of the system, knowing the identity of that public key or the person that key belongs to is essential. Given the asymmetric key architecture, anyone who can use the private key that corresponds to the public key will be authenticated and be allowed to use the association. But how do you tell who owns or has access to that key? The possibilities are mind-boggling. A particular key in a “To Account” authorized key file on a particular “To System” could be related to any “From Account” on any “From System” in the network.

The logical first step is to address the default implementation. This means examining every computer in the network and looking under every user’s home directory within their ~/.ssh directory to locate an identity file containing a matching public key. That is a huge effort, but even that isn’t enough. Users may have overridden the defaults and stored their identify keys files elsewhere in the file system. A zealous Information Security engineer might try to scan ~/.ssh/config files for non-default IdentityFile parameters with non-default directories and file names.

Secure SSH: Risk Management Solutions

However, even this is not comprehensive because identity files can be specified dynamically with the SSH client at runtime, and there is no way to easily track that. In other words, it may not be practical to look beyond the default locations. This is not ideal because the risk is probably greater for non-default identity files. A malicious user who has access to sensitive SSH keys, someone with the keys to the kingdom, would want to conceal that fact and is not likely to use the default IdentityFile. It would be very difficult, however, to track down these non-default locations that could include portable media such as a USB thumb drive.

Controlling Authorized_Keys Files

If the identity side of the key association is challenging, at least the authorized keys component is easier to control. The only way to prevent account owners from setting up new associations on target systems is to move the authorized keys file out of users' home directories and into a directory with controlled permissions. A centralized directory under /etc such as /etc/ssh/authorized_keys is often used. If this directory and the files contained within it are properly secured (e.g., owned by root and writeable only by root), then account owners will not be able to access the file to set-up up new key associations on their own. The sshd configuration parameter AuthorizedKeysFiles can be set to the location of a centralized directory, such as /etc/ssh/authorized_keys/%u. In this example, each user would have a file called /etc/ssh/authorized_keys/<userID> where authorized public keys are stored.

Centralized Provisioning of SSH Keys and Associations

The restrictive security permissions can help support centralized provisioning controls over the creation of key associations. For instance, the Information Security department can be responsible for provisioning key associations after obtaining proper approvals and documentation. Since key associations provide access to systems and can subvert existing security structures, it only makes sense that standard security provisioning controls be applied to requests for new key associations. It is only by centralizing the provisioning of new key associations through standard processes (including formal request and approval by account owners) that SSH keys can be controlled.

Secure SSH: Risk Management Solutions

Restricting SSH Connections via the Authorized Keys File Options

Besides listing the keys that are authorized to connect as each particular user, the authorized keys files can also define additional authorization options that can further restrict keys. These options can be quite useful for controlling and preventing abuse or misuse of keys that are authorized to connect to system accounts or accounts used for Secure FTP (SFTP) connections. Authorization options can be defined for each key and support a high degree of key-specific granularity. Some of the more useful options are described below:

- command="<some_predefined_command>" – Whenever this key is used to connect, this forced command (and only this command) is executed. Any commands requested by the client will be ignored. This option is useful for connections that provide single operations, such as backups, and nothing else.

- from="<list_of_authorized_host_names_or_IP_addresses>" - Only connections from the specified client hosts are permitted for this particular key. To connect, users must have the private key and connect from this designated source location. This makes it harder for an unauthorized user to use a stolen key. This option can be used when one system account has to connect to another system account. Restricting the connection to specific production servers can help limit the risk of abuse.

- no-pty – Prevents the allocation of a pseudo-terminal for interactive login sessions. Requests for an interactive login session will fail. Clients can connect with the key only to execute non-interactive commands or copy files between systems using the SCP command.

Restricting SSH Connection via Server-wide Configuration Parameters

Although not providing the same level of granularity, some sshd_config parameters on the target system can be used to provide limited server-wide restrictions on permissible key associations. By default, any account on the system can receive connections on an SSH server. Two server-wide configuration settings, which are rarely used, can restrict access to SSH connections using rules based on "To Accounts" and "From Systems." (Note that a "From Account" cannot be used in rules since the SSH servers have no idea of the "From Account's" identity, only that he has the corresponding private key to authenticate.)

The AllowUsers option permits only the specified “To Accounts” to receive inbound SSH connections; connections to all other accounts are blocked. Wildcards can be used in account names (e.g., ? = a single character and * = any sequence of characters). Accounts can also specify a host name, which is actually the name of the “From Server”, following the “@” symbol. The setting “AllowUsers timk@secureit.com” means that local user timk is allowed to receive connections from the remote server secureit.com. This would have the same effect as the “from” option in the AuthorizedKeysFile.

There is also a DenyUsers setting which denies access to all specified users and permits all others. If both DenyUsers and AllowUsers are defined, then any user that is denied will be blocked and only users that are allowed are permitted.

There are also AllowGroup and DenyGroup parameters available that work in similar ways based on group memberships. In some circumstances, these configuration options may be helpful. However, due to the lack of granularity, these server-wide configurations are often not adequate for effectively securing key associations.

The PermitRootLogin parameter, however, is frequently useful. The default value “prohibit-password” allows remote root logins via SSH using key-based authentication. A value of “no” will block any connections directly to the root account over SSH. This will require users to first connect as a non-privileged account. They may then use su or sudo to run commands as root. This provides an audit trail of the user assuming root’s identity. A value of “without-password” allows key-based authentication to root, which can present some problems, as will be seen later.

Tracking and Reviewing Key Associations

If an organization has a legacy SSH implementation with hundreds of thousands or even millions of key associations already in existence, it isn’t enough to merely implement a controlled provisioning process to manage new associations. The provisioning process will provide controls over new associations that will be added to the environment, but something still has to be done about the existing inventory of associations that have accumulated over the years. If OpenSSH operates in the default “self-serve” mode for several years, the resulting inventory of key associations is likely to contain a large number of inappropriate, unauthorized, or otherwise risky associations. Identifying these inappropriate key associations and removing them is key to cleaning up and securing the SSH environment.

Secure SSH: Risk Management Solutions

Even after the initial cleanup has been completed, additional tracking and reviewing of key associations should be performed. Effective security requires periodic review and recertification to ensure the entitlements are appropriate and continue to be required and authorized. Due to Sarbanes-Oxley (SOX) and other regulatory requirements, most organizations have implemented quarterly and semi-annual access recertification processes. Ideally, reviews of SSH key associations should fit into those existing recertification cycles.

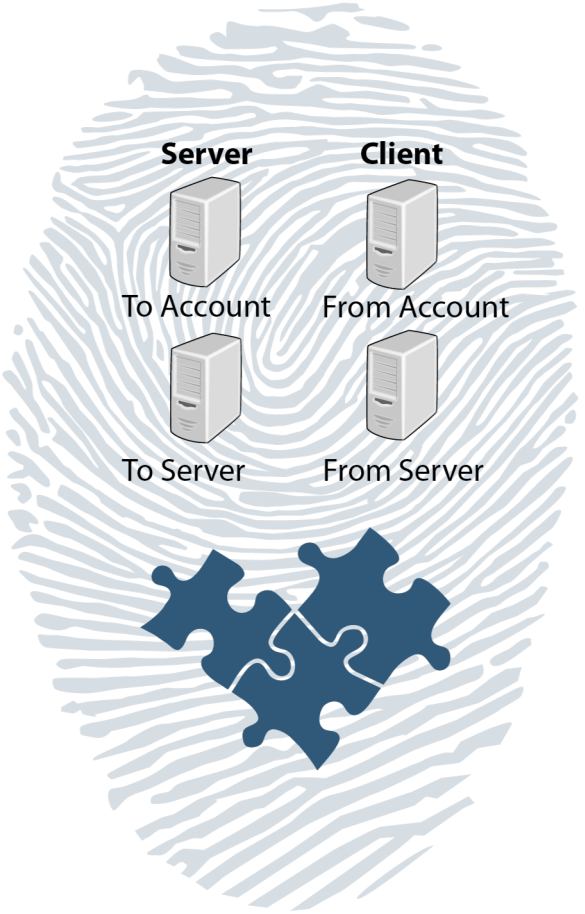
In order to validate the existing inventory of SSH key associations and recertify on an ongoing basis, organizations need a process to discover key associations. Most organizations perform this key association discovery task via scripts that run regularly on all systems in the environment. There are usually three steps to the process:

1. The first step is to extract information about the public keys that are stored in users' AuthorizedKeysFiles on the SSH servers. After files are centralized within a single directory (e.g., /etc/ssh/authorized_keys) on each server, it requires minimal effort to pull this information from systems. Rather than track the full public keys (which can be quite long), it is often preferable to track public key fingerprints which are MD5 hashes of the public key files. At this step of the discovery process, the following information is captured in a repository of key associations: "To Account," "To Server", and fingerprints of the authorized public keys. If additional authorization options are defined in the authorized keys files (e.g., forced commands, from restrictions, or no-pty), then it would be useful to capture those options as well. This information should be documented for periodic review and validation.

"Effective security requires periodic review and recertification to ensure that the entitlements are appropriate and continue to be required and authorized.

Secure SSH: Risk Management Solutions

2. The second step is to capture information about the identity of the public keys on the SSH clients. At a minimum, this requires scanning every user's home directory on every system for the default identity files. Keep in mind that an SSH client can be run on virtually any system (server or workstation) in the network. In addition, user home directories can be scattered throughout the file system. This is a lot of information to scan! After an identity file is found, the script typically generates a public key signature file and determines the user the key belongs to by identifying the owner of the home directory in which the identity file was located. Through this step of the discovery process, the following information is tracked: "From Account", "From Server", and fingerprints of public keys within the identity files. As mentioned previously, user identity files can be stored in directories other than the default location. Therefore, it is important to examine user-level ~/.ssh/config files (also stored in user home directories) to identify any non-standard locations. Including user-defined IdentityFiles parameters within the discovery process will increase the coverage of keys, but this will not detect all SSH keys in the environment because run-time options can be used to specify other directories. As mentioned earlier, it is not practical to identify every single SSH private key identity in the network, especially if a malicious user is determined to hide them. Using the authorization options on keys can help to manage the risk of an unidentified identity. To be absolutely sure that there is no risk, it is necessary to delete the association with the old keys and generate new keys. Periodic re-generation of keys is an accepted security practice for helping to protect keys from unauthorized use.



3. The third step of the discovery process is to correlate the information to produce a complete mapping of the key associations. Both the authorized key information and the identity file information include the public key fingerprint, which serves as a key field that can be used to join the “From” information (from the SSH client) and the “To” information (from the SSH server). The resulting information defines a complete key association trust relationship (“From Account,” “From Server,” “To Account,” and “To Server”) for a public key fingerprint.

Key Association Registry

Usually, information about the key association is stored in a registry for analysis. This registry can be integrated into the provisioning process so that new associations are added to the registry as they are created in the environment. It is also possible to re-scan the environment through the key discovery process described above on a regular basis and rebuild the key association registry for analysis purposes. Once complete, the registry of key associations will be critical in assessing the security risk of SSH key-based authentication and managing the security of the SSH environment.

Typical and recommended uses of an SSH key association registry to provide effective SSH key management include the following:

- Analyzing the existing inventory for risky or unauthorized associations that should be removed. This analysis can be based on high-level categories of associations. For example, any association that allows a user account to connect to a system account (an individual “From Account” and system “To Account”) presents risk of unauthorized access and should be avoided. Risks of these and other categories of key associations are discussed in the next eBook. Analysis can also be performed for specific accounts. For instance, if the organization has highly sensitive systems, system accounts for those systems warrant additional scrutiny and review.

- Performing periodic recertification of associations to validate their continuing need and appropriateness. The “To Account” owner should review the associations and validate that they are still required and appropriate. Given the likely volume of associations, it may be appropriate to focus the recertification only on those that present risk. Therefore, key associations where the “To Account” is identical to the “From Account” may not require recertification. These associations with matched accounts present minimal risk because users are allowed to access only their own accounts.

Secure SSH: Risk Management Solutions

- **Reviewing new associations within authorized keys files that have bypassed the authorized provisioning process.** Moving the authorized keys file under /etc prevents most users from provisioning keys, but system administrators (or others with access to root) could still access these files. Reconciling the associations within the authorized keys files against the registry of authorized associations that went through the provisioning process may help identify any rogue backdoors that were created directly by root users. Doing this sort of validation serves as a check on administrative users to guarantee that no associations bypassed the authorization and provisioning process.

- **De-provisioning keys and key associations when users terminate or access is no longer needed.** When employees leave the organization or system accounts no longer require access, good security practice mandates that keys/associations be disabled to prevent misuse. At a minimum, all key associations where the “From Account” belongs to a terminated user or retired system account should be removed. This is achieved by logging on to each “To Server” where the association is defined, accessing each “To Account’s” authorized keys file, and removing the public key that matches the “From Account’s” key fingerprint. After all the associations have been removed from the authorized keys files, no additional action is necessary, although the private keys could be removed from the user’s home directory on the “From System”. Usually, however, the user’s home directory will be removed via other de-provisioning activities, and this will consequently delete the private key from the environment (assuming the user’s keys were stored there).

- **Identifying key associations involving keys with unknown owners.** In all likelihood, some of the public keys that occur within authorized keys files on SSH servers will not be found through the scanning of SSH client identity files. Although these keys are authorized to connect through SSH authorized keys file, the owners of the keys have not been identified and no information exists regarding who can use the keys to connect. These unknown keys can occur due to several circumstances. For instance, keys could be stored in non-standard locations that were not scanned in the discovery process, or the keys may have been deleted and may no longer be accessible. Although it is difficult to determine exactly how to interpret these unknown keys, an excessively high number of them could indicate that the key discovery process and key association registry are not working effectively. In addition, associations with unknown keys could represent significant risk. Since the owners and users of these keys are unknown, it is not possible to determine if these associations introduce risk through unauthorized access. These associations warrant additional investigation and review. For high-risk “To Accounts,” keys with unknown owners should be carefully monitored to determine if they are being used. If not, then these key associations should be removed.

- **Rotating keys for risky associations.** Periodic rotation of encryption keys, which is basically creating new keys to replace older ones, is a generally accepted practice to maintain the security of cryptography over time. The registry of all key associations in the environment provides the necessary information to enable key rotation. Newly generated keys would need to be placed in the "From User's" home directory on each of the "From Servers." Then that key would need to be installed in the authorized keys files of the "To Users" on each of the "To Servers." As can be imagined, rotating SSH keys can require a significant amount of work. Therefore, it may not always be feasible to adopt a one-size-fits-all approach to key rotation. It may be acceptable to apply a tiered approach to key rotation, whereby only the highest risk keys are periodically regenerated and others keys may have longer lives.

It should also be noted that the key generation utility of OpenSSH supports an optional comment field that can be used to store the identity of the user to whom the key belongs and information such as a change request number. If keys are centrally provisioned by Information Security, this comment field can be used to track the identity of keys that are stored in the Authorized Keys File itself.

Conclusion

This eBook has outlined methods to properly secure and manage SSH, including key discovery, establishing a key association registry, and centralizing key provisioning. In our next eBook we will review the risks inherent in specific types of key associations that must be considered when creating new or validating existing associations, and provide suggestions for mitigating these risks.

Partnering with SecureIT

We hope you find this eBook helpful in your organization's path towards achieving a stronger security posture. As demonstrated above, SecureIT understands the operational practices and risks around network access but we also realize that no two organizations are alike. When SecureIT engages with our clients, we invest the time and resources to understand your organization, your software or services solution, and where you are in your journey.

Partnering with SecureIT to discover and mitigate security risks means ensuring that both your immediate and long-term compliance goals are achieved in an efficient manner so you can focus on the core job of increasing sales opportunities and growing revenue. Please contact us today, we would love to learn about your situation.

SECURE IT

About SecureIT

SecureIT provides risk, compliance, and cybersecurity services to enterprises, government entities, and cloud service providers. Our certified professionals assess cyber risk, conduct targeted security assessments, and ensure compliance with regulatory requirements. Every day, we partner with our clients to deliver solutions critical to protecting and growing business.

12110 Sunset Hills Road
Suite 600
Reston, VA USA 20190
703.464.7010
www.secureit.com