Secure SSH: Policy & Prevention

Greg Kent
Vice President, SecureIT

SECURE IT

Secure SSH: Policy and Prevention

Introduction

This eBook covers the risks inherent in specific types of key associations that must be considered when creating new or validating existing associations and provides suggestions for mitigating these risks. This third and final eBook on Secure SSH wraps up with recommendations for enterprise-wide standardized SSH configurations, including specific SSH server settings.

Policy Control to Address Risks of Key Associations

The risks of key associations need to be considered when a new association has been requested, as well as in analyzing and validating existing "legacy" associations that have accumulated over the years. Although each association presents its own unique risks, there are broad categories of associations that share common themes of risk and have implications for control. It is advisable for organizations to make policy decisions about these categories of key associations, defining which are acceptable and which are not. In most cases, at least some of the key associations should be prohibited by policy. The risks and along with suggested approaches for handling these associations are discussed in table starting on the next page.

> *"…the default OpenSSH implementation follows a self-serve key management model that can introduce security risk and result in inappropriate access through the use of key associations.*

Secure SSH: Policy and Prevention

System-Based Risks

Type of Association	Discussion of Risk and Other Concerns
Person Account to System Account	This association is risky since it allows people to assume the identity of system accounts. Passwords for system accounts are generally tightly controlled to prevent abuse and misuse of system accounts. Access through SSH key associations can circumvent these controlled processes and allow unauthorized users to obtain/retain access. This is especially a risk if security associations are not well-controlled (e.g., validated from time-to-time) and private keys not well protected. These associations should be avoided.
System Account to Person Account	It is hard to imagine the circumstances under which such a key association would be required. Systems accounts should only be used for production processes and these should never perform functions in the security context of individual person accounts. These associations can also be used for transitive attacks. For example if PersonA can associate with System1 and System1 can associate with PersonB, then essentially PersonA can connect as PersonB. Transitive relationships can get very complicated very quickly, so it is best to avoid them. The ideal solution is to segregate Person Accounts and System Accounts and avoid any associations between those two categories.

Secure SSH: Policy and Prevention

Type of Association	Discussion of Risk and Other Concerns
System Account to same System Account	Since both the “From Account” and the “To Account” are the same system account used by automated processes, the risk of these associations is negligible. In many cases, a centralized authentication repository (such as LDAP) provides a single authentication source throughout the environment. In this case, there is no risk for key associations with the same account name. However, any association where the “From Account” is a system account potentially presents risk in terms of protecting the keys. Keys used by system accounts often do not have passphrases defined. The risk and possible solutions for this issue were discussed in the first eBook in this series (e.g., using an ssh agent). In addition, appropriate authorization options to limit commands, source addresses, and interactive logons should also be considered since these options can greatly minimize the impact if a key is stolen.
System Account to different System Account	Since both the “From Account” and the “To Account” are system accounts used by automated processes, the risk of these associations may be acceptable, provided that there is a legitimate business need for the association.
Any Account to Root Account	Any account, system or person that can connect to the root account introduces risk. Any remote connections to root should be avoided. In keeping with good security practices, systems should require users to connect locally as a lower-privileged account and then SU to root in order to provide accountability and audit trail. There is an sshd_config parameter that will disallow SSH connections to root. A setting of “PermitRootLogin no” will prevent all root logons. Settings of “yes” or “without-password” allow connections via ssh keys, and should be avoided.

Secure SSH: Policy and Prevention

Type of Association	Discussion of Risk and Other Concerns
Person Account to same Person Account	When key associations are between like accounts (e.g., To Account is FredW and From Account is FredW), there is little risk from the association itself since users can only connect to their own accounts on different systems. However, person-to-person associations can be a potential headache because there can be a lot of these associations. A single user might want to be able to connect to dozens or even hundreds of systems throughout the network, which means a new key association must be setup for each target system--a significant burden on the provisioning operation. In addition, there are other risks and challenges for keys assigned to person accounts, such as users compromising the security of their private keys by having keys without passphrases or sharing keys with other users. If privacy of the private key cannot be assured, then key-based authentication may be less reliable than traditional password controls. At least passwords are generally stored securely on the system and are not susceptible. For this reason, some organizations decide to disallow person-to-person associations and require users to connect to SSH using password authentication. In this model, key associations are permitted only for system accounts, which can be more easily managed and controlled. If the number of person associations is not overwhelming, the Information Security provisioning function could create the keys and use high-quality passphrases during key generation.
Person Account to different Person Account	By definition, these associations allow one human user to assume the identity of another human user. This causes a loss of accountability and undermines access controls. These types of associations should never be permitted.

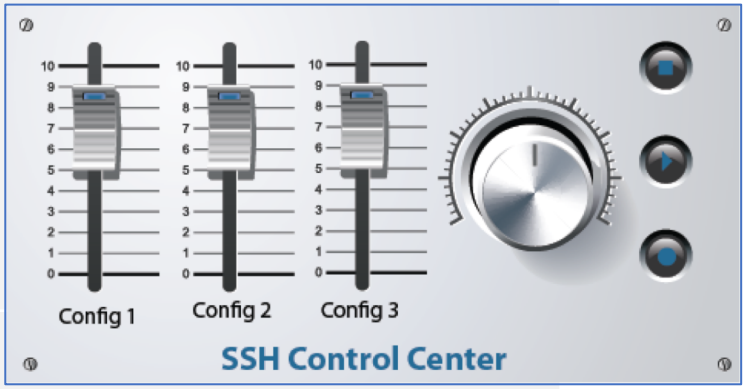
Secure SSH: Policy and Prevention

Type of Association	Discussion of Risk and Other Concerns
Multiple “From Accounts” (Person or System) for the same key	Use of the same key by multiple “From Accounts” indicates that the key has been shared and that the privacy of the private key has been compromised. At a minimum, it is necessary to remove all known instances of “From Accounts” with unauthorized keys. That is, delete the private key from the home directories of all “From Accounts” whose logon ID does not correspond to the “To Account.” However, since the key was compromised, this approach does not address the risk that unauthorized copies of the private key may be stored elsewhere. The safest approach is to delete all instances of the key, remove the key association entirely, and generate a new key and key associations. After the old association is removed, no one with an unauthorized copy of the private key can connect. Then create a new key that can be protected from the beginning and create new key associations that are required so that each “From Account” uses a different key.
Unknown Keys occur in the authorized keys 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. 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 of unauthorized access. It is possible, for example, that some of the unknown keys may exist because malicious users have intentionally hidden the private key in order to preserve their access to system accounts. Because of the potential risk, the unknown keys should be removed from the AuthorizedKeysFiles for system accounts. However, there is also a Denial of Service (DoS) risk with removing these associations since the keys could also be used by system “From Accounts” that are required. Therefore, careful analysis is required before associations are removed. Removal of associations of unknown accounts to person “To Accounts” are relatively more difficult to detect since there would be no impact on production and users could still authenticate through passwords. Associations to system “To Accounts” may require extensive logging in order to identify if the unknown public keys are dormant or are still being used. Obviously, if unknown public keys have not been used for an extended period of time, they can be safely deleted without impacting any processing.

Secure SSH: Policy and Prevention

System-Based Risks

Type of Association	Discussion of Risk and Other Concerns
Low Risk Systems to High Risk Systems	Some organizations have different levels of control applied to different systems. For instance, systems supporting financially significant applications may be subject to tighter restrictions and control practices than systems related to lower risk non-financial applications. Or systems in a testing environment may be more loosely secured than systems in production. If this is the case, then associations between these categories of systems should be avoided. If low risk systems can be compromised more easily by attackers or provide greater access to humans, these systems could be the means through which attacks or unauthorized access spreads to higher-risk systems.
External Systems to Internal Systems	Although session encryption capabilities of SSH seem ideally suited to connections that span across networks, there are risks associated with keys that need to be considered. Once keys are sent outside the network, they are no longer in the control of the organization. Practices that might be enforced internally may not be enforced outside the network. If a key is compromised from an external system, then internal systems may be at risk. Reasonable measures should be taken to help protect keys from compromise (such as generating keys internally with strong phrases and securely distributing them to users outside the network). If inbound SSH sessions cannot be completely blocked at the perimeter, then at least firewall restrictions on the source and destination of SSH connections should be used to restrict access as much as possible.



Secure SSH: Policy and Prevention

Standardized Configuration

A final element of effective management of SSH keys and trust relationships is having standardized SSH configurations across the environment. Configurations can be defined for both the SSH client and the SSH server. As mentioned earlier in previous eBooks, the configuration of the SSH client can be quite challenging, since users can override any standard configuration in the OpenSSH model. Fortunately, there are relatively few highly significant configuration settings on the client side of SSH. The IdentityFile parameter (as discussed earlier) and the StrictHostKeyChecking parameter (which concerns SSH host keys and is beyond the scope of this eBook) are the most significant settings.

There are several SSH server configurations in the /etc/ssh/sshd_config file that are significant and should be assessed. The typical controls for managing configurations should be deployed, including baseline configuration standards/builds and periodic validation of server settings in production to detect non-compliance and configuration drift. Some of the critical settings that should be assessed are described in the table below:

sshd_config Parameter	Defaults	Recommended Value
AuthorizedKeysFile – can be used to store authorized keys files in a centralized directory that blocks user update access, thereby providing enforcement for an authorized provisioning process	~/.ssh/authorized_keys	/etc/ssh/authorized_keys/%u or some other central directory outside of users' home directories and properly secured to deny write access
Ciphers – used to define the ciphers and key lengths that are supported	Version 2 only: aes128-cbc, 3des-cbc, blowfish-cbc, cast128-cbc, arcfour128, arcfour256, arcfour, aes192-cbc, aes256-cbc, aes128-ctr, aes192-ctr, aes256-ctr. For Solaris: aes128-ctr, aes128-cbc, 3des-cbc, blowfish-cbc, arcfour	Default or only 128+ bit ciphers, or stronger if required
HostBasedAuthentication – can be used to refuse or enable host-based authentication	no	Default or no (If set to yes, then IgnoreRhosts should be yes)

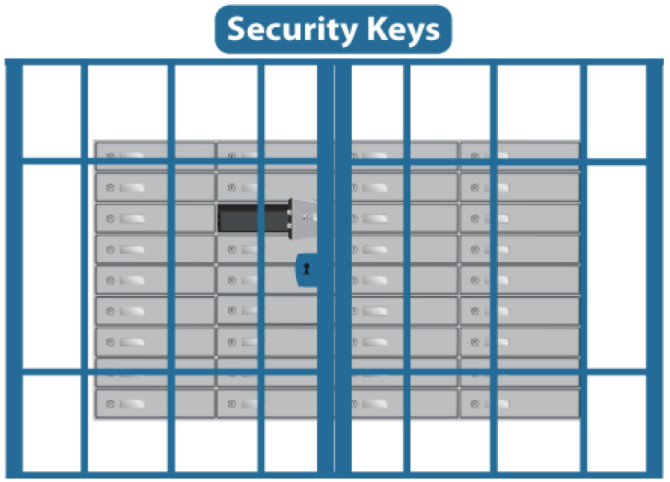
Secure SSH: Policy and Prevention

sshd_config Parameter	Defaults	Recommended Value
IgnoreRhosts – can be used to disable used of .rhosts and .shosts files	yes	Default or yes (This has no affect if HostBasedAuthentication is no)
LogLevel – can be used to specify the level of logging that is performed by the SSH server	INFO	Default or INFO or VERBOSE Note: verbose may generate significantly more log data
MaxAuthTries – prevents password brute forcing by setting a cap on failed attempts	6	Default or 6 or less
PermitEmptyPasswords – can be used to disallow empty password	no	Default or no
PermitRootLogin – can be used to prevent access to the root account	yes	no

Notes:
* Setting the configuration option to prevent direct access to the root account via keys helps to avoid the risk of unauthorized key associations for root access.
* If HostBasedAuthentication is used, then there are two things to consider. First, each host must have a unique host key (e.g., multiple hosts should not share the same key). SSH uses host keys to validate the identity of systems, so sharing a host key makes it easier to defeat the host authentication process. If host authentication is used, then host keys should be reviewed to make sure that keys are not shared across systems. Second, IgnoreRhosts should be enabled. Host based authentication uses the standard "r" services trust files (/etc/hosts.equiv and ~/.rhosts) and SSH-specific trust files (/etc/shosts.equiv and ~/.shosts) to define the trust relationships that are permitted. Essentially, these 4 files define the same sort of trust associations that are defined in the authorized keys file for user keys. Users will be able to access the ~/.rhosts and ~/.shosts files in their home directories, and can therefore create new trust relationships at will. In order to lock-down the trust relationships so that they can be properly controlled, SSH must be configured to ignore those trust files that are directly accessible to the end user, which is what the IgnoreRhosts option does. When this option is set, then all trust relationships must be defined within either the hosts.equiv and shosts.equiv files, both of which are locked-down under the /etc directory. This approach eliminates "self-serve" trust relationships and enables host based trust relationships to be managed and controlled in a similar way as is described in this paper for user keys.
* Default values on OSX maybe different than the values listed here.

Secure SSH: Policy and Prevention

sshd_config Parameter	Defaults	Recommended Value
Protocol – can be used to only allow specific SSH protocol versions to be used	2,1	2 [Version 1 is susceptible to some security attacks.]
PubKeyAuthentication – can be used to enable public keys	yes	Default or yes
StrictModes – can be used to ensure that the SSH server program will not execute if the authorized keys file and home directory is insecure	yes	Default or yes
UsePrivilegeSeparation – can be used to prevent privilege escalation attacks by spawning least-privileged processes	yes	Default or yes



Conclusion

SSH is a widely deployed facility for establishing encrypted connections. Although it was intended to solve a security problem, the default OpenSSH implementation follows a self-serve key management model that can introduce security risk and result in inappropriate access through the use of key associations. In order to properly secure and manage SSH, traditional controls need to be applied to key associations, including centralized provisioning and de-provisioning, periodic recertification, and centralized tracking of user entitlements.

Secure SSH: Policy and Prevention

To accomplish these objectives, it is necessary first to centralize authorized keys files into a directory that is not accessible to users. Then it is useful to scan the environment and aggregate information about authorized keys on SSH servers and user identities on SSH clients. A registry of SSH key associations can provide valuable information to help identify inappropriate or otherwise risky associations that are defined in the environment. These reasonable steps can help organizations that do not have a formal PKI infrastructure to manage and control the deployment of OpenSSH keys and key associations. Since key-based authentication is critical feature of SSH, effective key management is required in order to have a reliable authentication mechanism and prevent keys from being used to circumvent controls that exist for password-based logons. More fundamentally, key management is needed so organizations can restrict system access to known, authorized users.

For more information on SSH key management, see the draft IETF Best Current Practice (BCP) co-written by SecureIT at https://www.ietf.org/archive/id/draft-ylonen-sshkeybcp-01.txt

Partnering with SecureIT

We hope you have found SecureIT's eBook series on Secure SSH 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