Security requirement

Secure Shell (SSH)

Deutsche Telekom Group

Version	4.0
Date	Jul 1, 2024
Status	Released

Publication Details

Published by
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Germany

File name	Document number	Document type
	3.04	Security requirement
Version	**State**	**Status**
4.0	Jul 1, 2024	Released
Contact	**Validity**	**Released by**
Telekom Security psa.telekom.de	Jul 1, 2024 - Jun 30, 2029	Stefan Pütz, Leiter SEC-T-TST

Summary
Security requirements for SSH server, SFTP server and the SSH protocol.

Copyright © 2024 by Deutsche Telekom AG.
All rights reserved.

Table of Contents

1. Introduction 4
2. sshd_config 5
3. permissions & key material 15

1. Introduction

SSH (Secure Shell) is a client/server application inclusive a network protocol that can be used to access systems on terminal level and to transfer data. As SSH is typically used for management access, this service has a high security demand. This document includes security requirements for SSH server, SFTP server and the SSH protocol.

It is recommended to use OpenSSH as this is a well-known solution with a huge developer community. Accordingly requirements are aligned to OpenSSH.

This security document has been prepared based on the general security policies of the group in correlation with security best practices and vendor recommendations. The security requirement is used as a basis for an approval in the PSA process, among other things. It also serves as an implementation standard for units which do not participate in the PSA process. These requirements shall be taken into account from the very beginning, including during the planning and decision-making processes. When implementing these security requirements, the precedence of national, international and supranational law shall be observed.

2. sshd_config

Req 1	The SSH protocol version 2 must be used. (Automatically compliant when using OpenSSH 7.4+ .)

SSHv1 must permanently be disabled in configuration of the SSH server. With OpenSSH 7.4 support for SSHv1 completely removed and must not longer be configured.

Motivation: SSH protocol version 1 has weaknesses and is obsolete today. Because it is vulnerable to Man-in-the-Middle attacks confidentiality and integrity of transmitted data cannot be guaranteed.

Implementation example: For OpenSSH Versions supporting it `sshd_config` contains:

```
Protocol 2
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-1/4.0

Req 2	SSH MaxStartups must be set to "10:30:100" or less. (OpenSSH default)

The MaxStartups parameter specifies the maximum number of concurrent unauthenticated connections to the SSH daemon. This value must be "`10:30:100`" or less, this is OpenSSH default.

Motivation: To protect a system from denial of service due to a large number of pending authentication connection attempts, use the rate limiting function of MaxStartups to protect availability of sshd logins and prevent overwhelming the daemon.

Implementation example: `sshd_config` contains:

```
MaxStartups 10:30:100
```

For this requirement the following threats are relevant:
- Disruption of availability

ID: 3.04-2/4.0

Req 3	Only approved key exchange algorithms must be used. See longtext for a complete list.

For key exchange the following algorithms are allowed:
- sntrup761x25519-sha512@openssh.com
- curve25519-sha256@libssh.org
- curve25519-sha256
- ecdh-sha2-nistp521
- ecdh-sha2-nistp384
- ecdh-sha2-nistp256
- diffie-hellman-group18-sha512
- diffie-hellman-group16-sha512
- diffie-hellman-group-exchange-sha256

Motivation: Attacks could break the encryption of transported data if weak ciphers and algorithms are used.

Implementation example: `sshd_config` contains:

```
KexAlgorithms sntrup761x25519-sha512@openssh.com,curve25519-sha256@libssh.org,curve25519-sha256,ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group18-sha512,diffie-hellman-group16-sha512,diffie-hellman-group-exchange-sha256
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.04-3/4.0

Req 4	Only approved ciphers algorithms must be used. See longtext for a complete list.

Outdated and insecure ciphers and algorithms must not be used. Use the following ciphers for SSH:

- aes256-gcm@openssh.com
- aes128-gcm@openssh.com
- chacha20-poly1305@openssh.com
- aes256-ctr
- aes192-ctr
- aes128-ctr

Motivation: Attacks could break the encryption of transported data if weak ciphers and algorithms are used.

Implementation example: `sshd_config` contains:

```
Ciphers aes256-gcm@openssh.com,aes128-gcm@openssh.com,chacha20-poly1305@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.04-4/4.0

Req 5	Only approved MAC algorithms must be used. See longtext for a complete list.

It is important to avoid the use of insecure MAC algorithms for SSH. Examples of such outdated algorithms are MD5 and SHA1. The following MAC algorithms are allowed and must be configured for SSH daemon:

- hmac-sha2-512-etm@openssh.com
- hmac-sha2-512
- hmac-sha2-256-etm@openssh.com
- hmac-sha2-256

Motivation: Attacks could break the encryption of transported data if weak ciphers and algorithms are used.

Implementation example: `sshd_config` contains:

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-512,hmac-sha2-256-etm@openssh.com,hmac-sha2-256
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.04-5/4.0

Req 6	Only approved Host Key Algorithms (a.k.a. public key signature algorithms or server authentication algorithms) must be used. See longtext for a complete list. A Host Key must identify a host distincly. (1:1 relation)

Outdated and insecure algorithms must not be used. Use the following host key algorithms. Naming can be different from software too software:

- ssh-ed25519
- ssh-ed25519-cert-v01@openssh.com
- sk-ssh-ed25519@openssh.com
- sk-ssh-ed25519-cert-v01@openssh.com
- ecdsa-sha2-nistp521
- ecdsa-sha2-nistp521-cert-v01@openssh.com
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp384-cert-v01@openssh.com
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp256-cert-v01@openssh.com
- sk-ecdsa-sha2-nistp256@openssh.com
- sk-ecdsa-sha2-nistp256-cert-v01@openssh.com
- rsa-sha2-512
- rsa-sha2-512-cert-v01@openssh.com
- rsa-sha2-256
- rsa-sha2-256-cert-v01@openssh.com

Signature algorythm `ssh-rsa` is considered insecure and therefore not allowed anymore. RSA key material with sufficient key size can still be used with RSA SHA-2 signature algorithms `rsa-sha2-256/512` .

Motivation: Malicious hosts could imitate a legitimate host undetected.

Implementation example: `sshd_config` contains:

```
HostKeyAlgorithms ssh-ed25519,ssh-ed25519-cert-
v01@openssh.com,sk-ssh-ed25519@openssh.com,sk-ssh-ed25519-cert-
v01@openssh.com,ecdsa-sha2-nistp521,ecdsa-sha2-nistp521-cert-
v01@openssh.com,ecdsa-sha2-nistp384,ecdsa-sha2-nistp384-cert-
v01@openssh.com,ecdsa-sha2-nistp256,ecdsa-sha2-nistp256-cert-
v01@openssh.com,sk-ecdsa-sha2-nistp256@openssh.com,sk-ecdsa-
sha2-nistp256-cert-v01@openssh.com,rsa-sha2-512,rsa-sha2-512-
cert-v01@openssh.com,rsa-sha2-256,rsa-sha2-256-cert-
v01@openssh.com
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.04-6/4.0

Req 7	SSH logging must be enabled.

Logging for SSH must be enabled. It is recommended to use level INFO to get important information but not to get a lot of useless events. If needed higher levels like VERBOSE can also be used.

Motivation: Logging security-relevant events is a basic requirement for detecting ongoing attacks as well as attacks that have already occurred. This is the only way in which suitable measures can be taken to maintain or restore system security. Logging data could be used as evidence to take legal steps.

Implementation example: `sshd_config` contains:

```
SyslogFacility AUTH
LogLevel INFO
```

For this requirement the following threats are relevant:
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-7/4.0

Req 8	SSH LoginGraceTime must be set to 120s or less. (OpenSSH default)

The LoginGraceTime parameter restricts the time window for a successful authentication. The longer this period is the more open unauthenticated connections can be established. Required is a maximum of 120s, this is OpenSSH default setting.

Motivation: An adequate time for LoginGraceTime parameter protects the system against unauthenticated SSH connections which waste system resources.

Implementation example: `sshd_config` contains:

```
LoginGraceTime 120
```

For this requirement the following threats are relevant:
- Disruption of availability

ID: 3.04-8/4.0

Req 9	SSH MaxAuthTries must be set to 6 or less. (OpenSSH default)

The MaxAuthTries parameter specifies the maximum number of authentication attempts permitted per connection. This value must be limited to 6 or less attempts, this is OpenSSH default.

Motivation: This parameter will minimize the risk of successful brute force attacks to the SSH server.

Implementation example: `sshd_config` contains:

```
MaxAuthTries 6
```

For this requirement the following threats are relevant:
- Disruption of availability

ID: 3.04-9/4.0

Req 10	SSH root login must be disabled.

Unique, personal user accounts are mandatory. Constantly working as root is not permitted. To avoid remote login with user root the login over SSH must be disabled.

Note: It is also possible to achieve an adequate security level if only functional user accounts are used on a system. It must be guaranteed to share SSH keys over a central account management system (e.g. ZAM) for the root user and to enroll them with a configuration management system. Additionally, access must be

done over a jump host with personalized accounts. The use of SSH keys for authentication is still mandatory (login with password over SSH is not allowed).

Motivation: Usage of root user for administrative tasks as daily routine is a risk, because root has unlimited rights, exists by default and therefore is an easy target for attacks. For everyday work individual, restricted accounts reduce risk. ("Least Privilege Principle")

Implementation example: `sshd_config` contains:

```
PermitRootLogin no
```

For this requirement the following threats are relevant:
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-10/4.0

Req 11	SSH strict mode must be enabled.

SSH StrictModes must be enabled. This enables checks to ensure that SSH files and directories have the proper permissions and ownerships of the login user before allowing an SSH session to open.

Motivation: Prevents usage of folder or files regarding OpenSSH with too open access rights.

Implementation example: `sshd_config` contains:

```
StrictModes yes
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-11/4.0

Req 12	SSH user authentication must be done with public keys.

Authentication with public/private key must be used for SSH login.

Note: The private key of human beings must be protected with a passphrase.

Motivation: Passwords are usually attackable via Phishing, Keylogger and Brute Force attacks. Generally passwords are easier to attack than private SSH keys, esspecially if passphrases are used for private SSH keys.

Implementation example: `sshd_config` contains:

```
AuthenticationMethods publickey
PubkeyAuthentication yes
```

For this requirement the following threats are relevant:
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.04-12/4.0

Req 13	SSH password authentication must be disabled.

The login must be done with public/key authentication. Login with password only must be disabled for SSH.

For newer Linux Distribution "`UsePAM yes`" can be necessesary to keep support and system working as expected. In this case PAM configuration must not bypass mandatory user authentication, regardless if it be single factor authentication for unprivileged user or be it 2FA for administrators.

Systems that are connected to a central IAM may require authentication by account name and password. In this case, use is only allowed for centrally managed accounts. This deviation can be realized, for example, by means of "`Match` " configuration, see "Match Introduces a conditional block". Privileged accounts of a central IAM can then implement 2FA authentication, for example through "`AuthenticationMethods password,publickey`".

Motivation: Passwords are usually attackable via Phishing, Keylogger and Brute Force attacks. Generally passwords are easier to attack than private SSH keys, esspecially if passphrases are used for private SSH keys.

Implementation example: `sshd_config` contains:

```
PasswordAuthentication no
KbdInteractiveAuthentication no
ChallengeResponseAuthentication no
UsePAM no
```

For this requirement the following threats are relevant:
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.04-13/4.0

Req 14	SSH IgnoreRhosts must be enabled.

Motivation: Usage of ".rhosts" file would allow login without a authentication feature like SSH private key.

Implementation example: `sshd_config` contains:

```
IgnoreRhosts yes
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-14/4.0

Req 15	SSH HostbasedAuthentication must be disabled.

Motivation: If a trust relationship is configured with another system a succesfull attack also means compromise of all other trusted systems.

Implementation example: `sshd_config` contains:

```
HostbasedAuthentication no
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data

- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-15/4.0

Req 16	The usage of the SSH service must be restricted to dedicated groups or users.

For easier and more secure system administration it is necessary to use dedicated users or groups (recommended) for SSH.

Motivation: The usage of dedicated users or groups makes use of SSH more secure through reduction of access rights to the necessary minimum. ("Least Privilege Principle")

Implementation example: `sshd_config` contains options like `AllowUsers`, `AllowGroups`, `DenyUsers`, `DenyGroups` to control which user or groups are allowed to login. See also manpage sshd_config.

For this requirement the following threats are relevant:
- Unauthorized access to the system

ID: 3.04-16/4.0

Req 17	SSH MaxSessions must be set to 10 or less. (OpenSSH default)

The MaxSessions parameter specifies the maximum number of open shell, login or subsystem (e.g. sftp) sessions permitted per network connection. This value must be 10 or less, this is OpenSSH default.

Motivation: To protect a system from denial of service due to a large number of concurrent sessions, use the rate limiting function of MaxSessions to protect availability of sshd logins and prevent overwhelming the daemon.

Implementation example: `sshd_config` contains:

```
MaxSessions 10
```

For this requirement the following threats are relevant:
- Disruption of availability

ID: 3.04-17/4.0

Req 18	SSH tunnel devices must be disabled.

SSH can be used to tunnel services. For management service of Linux servers this is typically not used and can be disabled.

Motivation: SSH tunnel feature can be used by an attacker to tunnel traffic to own destinations.

Implementation example: `sshd_config` contains:

```
PermitTunnel no
```

For this requirement the following threats are relevant:
- Unauthorized use of services or resources

ID: 3.04-18/4.0

Req 19	SSH TCP port forwarding must be disabled.

TCP forwarding can be used to forward TCP connections through SSH. For management service of Linux servers this

is typically not used and can be disabled.
Note: This requirement is not valid for Jumphosts !

Motivation: Bypassing firewall rule sets get possible, as TCP Forwading allows to tunnel over a SSH server.

Implementation example: `sshd_config` contains:

```
AllowTcpForwarding no
```

or better

```
DisableForwarding yes
```

For this requirement the following threats are relevant:
- Unauthorized use of services or resources

ID: 3.04-19/4.0

Req 20	SSH agent forwarding must be disabled.

SSH agent forwarding allows to forward authentication requests to other systems over SSH. For management service of Linux servers this is typically not used and must be disabled.
Note: This requirement is not valid for Jumphosts!

Motivation: The server-side deactivation blocks the creation of a server-side agent forwarding socket, this socket consequently cannot be misused.

Implementation example: `sshd_config` contains:

```
AllowAgentForwarding no
```

or better

```
DisableForwarding yes
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-20/4.0

Req 21	SSH gateway ports must be disabled.

SSH Gateway ports specifies whether remote hosts can connect to ports forwarded for the client. For management service of Linux servers this is typically not used and can be disabled.

Motivation: Bypassing firewall rule sets get possible, as GatewayPorts allows to tunnel over a SSH server.

Implementation example: `sshd_config` contains:

```
GatewayPorts no
```

or better

```
DisableForwarding yes
```

For this requirement the following threats are relevant:
- Unauthorized use of services or resources

ID: 3.04-21/4.0

Req 22	SSH X11 forwarding must be disabled.

X11 is not used on Linux servers. The forwarding of X11 over SSH must be disabled.

Motivation: If this feature is not used in a controlled manner, it could be a security risk for servers.

Implementation example: `sshd_config` contains:

```
X11Forwarding no
```

or better

```
DisableForwarding yes
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-22/4.0

Req 23	SSH PermitUserEnvironment must be disabled.

The SSH option `PermitUserEnviroment` specifies if user defined environment variables are processed by sshd. This function must disabled by setting option argument to "`no`".

Motivation: Enabling the processing environment variable may enable users to bypass SSH access restrictions.

Implementation example: `sshd_config` contains:

```
PermitUserEnvironment no
```

For this requirement the following threats are relevant:
- Unauthorized use of services or resources

ID: 3.04-23/4.0

Req 24	SSH PermitEmptyPasswords must be disabled.

With the "`PermitEmptyPasswords`" option can be configured the SSH server allows login to an account with an empty password. This must not be allowed.

Motivation: If login without a password remotely over SSH is possible unauthorized users can get access to the server.

Implementation example: `sshd_config` contains:

```
PermitEmptyPasswords no
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.04-24/4.0

Req 25	If SFTP is activated, internal server of OpenSSH must be used.

OpenSSH has its own SFTP daemon. If SFTP should be used this function must be enabled and configured in a secure way.

Motivation: It is necessary to use the OpenSSH SFTP daemon to align the security configuration for all SSH based services and not to have different security levels.

Implementation example: `sshd_config` contains:

```
Subsystem sftp internal-sftp
```

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized use of services or resources

ID: 3.04-25/4.0

3. permissions & key material

Req 26	SSH configuration files and SSH private keys must be protected with a mask of `0600` or more restrictive. Public keys need a mask of 0644 or more restrictive.

The `sshd_config` and files under `sshd_config.d/` contains configuration specifications for sshd. sshd usually runs as root and its configuration and key material (`ssh_host_*_key` and `ssh_host_*_key.pub`) must be protected appropriately.

Motivation: Configuration files and key material need to be protected from unauthorized changes by non-privileged users.

Implementation example:

```
$ chown root:root /etc/ssh/sshd_config
$ chmod 600 /etc/ssh/sshd_config
$ chown root:root /etc/ssh/sshd_config.d/*
$ chmod 600 /etc/ssh/sshd_config.d/*
$ find /etc/ssh -xdev -type f -name 'ssh_host_*_key' -exec chown
root:root {} \;
$ find /etc/ssh -xdev -type f -name 'ssh_host_*_key' -exec chmod
600 {} \;
$ find /etc/ssh -xdev -type f -name 'ssh_host_*_key.pub' -exec
chown root:root {} \;
$ find /etc/ssh -xdev -type f -name 'ssh_host_*_key.pub' -exec
chmod 644 {} \;
```

For this requirement the following threats are relevant:
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.04-26/4.0

Req 27	If RSA keys are used, they must be key size 3072 bit or more.

Available compute power and new attack methods make it necessary to review used key material and encryption algorithms for secure data exchange. This is especially true for key material that is rarely or never rotated.
For RSA a key size of 3072 bit or more is necessary. If a key change is necessary, the recommendation is to switch to ED25519 hostkeys.

Motivation: RSA keys of small size could allow brute force attacks to be succesful.

Implementation example: RSA:

```
$ ssh-keygen -t rsa -b 3072 -f /etc/ssh/ssh_host_rsa_key
```

ED25519:

```
$ ssh-keygen -t ed25519 -f /etc/ssh/ssh_host_ed25519_key
```

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unnoticeable feasible attacks

ID: 3.04-27/4.0

Req 28	SSH moduli equal to or greater 3071 must be used.

The file "/etc/ssh/moduli" contains pre-generated group parameters – named moduli - for Diffie-Hellman. Here are

also moduli available that are not long enough to withstand known attacks. To avoid the use of short values moduli smaller than 3071 (respectively 2047 for existing systems) must be deleted from file " /etc/ssh/moduli ".

Motivation: If the DH moduli is to short the key exchange is not protected in an adequate way.

Implementation example:

```
$ awk '$5 >= 3071' /etc/ssh/moduli > /etc/ssh/moduli.safe
$ mv /etc/ssh/moduli.safe /etc/ssh/moduli
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data

ID: 3.04-28/4.0