Security requirement

Linux OS for Servers

Deutsche Telekom Group

Version	8.0
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
	3.65	Security requirement
Version	**State**	**Status**
8.0	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 Linux OS for servers inclusive requirments for iptables, mandatory access control solutions etc.

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

Table of Contents

1.	Introduction	4
1.1.	Scope	4
1.2.	Not in Scope	4
2.	System Hardening	5
2.1.	Network Hardening	12
2.2.	System Update	14
2.3.	User Authentication	18
2.4.	Administrative Access	21
2.5.	Location	22
3.	Logging	24
3.1.	Auditd	26
3.2.	External Logging	29
3.2.1.	RSyslog	31
3.2.2.	Syslog-NG	31
4.	Authentication attribute "password" & Pluggable Authentication Modules	33
5.	host-based firewall	38
6.	Mandatory Access Control (MAC)	40
6.1.	SELinux	40
6.2.	AppArmor	42
7.	Regular Compliance Checks	44
8.	Technical Baseline Security for IT/NT Systems	48
8.1.	Basic System Hardening	48
8.2.	Protecting availability and integrity	48
8.3.	Protecting data and information	49
8.4.	Session Protection	50
8.5.	Authentication and authorization	51
8.6.	Authentication attribute "password"	51
9.	Operating Systems	55
9.1.	System Hardening	55
9.2.	System Update	55
9.3.	Availability and Integrity	56
9.3.1.	TCP/IP Stack	56
9.4.	Authentication and Authorization	58

1. Introduction

Linux is a popular operating system for server systems. It is used from single bare metal servers to high scaling cloud environments. This document includes security requirements for the base operating system and also for applications which are part of the operating system like, iptables, mandatory access control solutions etc. The security requirements are valid for all Linux distributions.

The security requirements have 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.

A tool for hardening a Linux system according to requirements in this docuement can be found here.

1.1. Scope

In scope of this is document are server systems with Linux OS. This includes the following types of servers:

- bare metal servers
- virtualized servers (VMs)
- virtualization hosts

1.2. Not in Scope

This document is not intended to be used for workstations with Linux OS and container images based on Linux. Because of different use cases, associated threats and needed software, workstations need another set of requirements. Container images have to be treated differently as they normally do not include a full Linux OS.

2. System Hardening

Req 1	Unnecessary services must be disabled.

After the installation of systems and software products, supplier-preset, local or network-accessible services are often active that are not required for the operation and functionality of the specific system in the intended operating environment.

However, in principle only the services actually required may be active on a system.

Accordingly, all services that are not required on a system must be completely disabled immediately after installation. It must be ensured that these services remain disabled even after the system is restarted.

Motivation: Active services that are not required unnecessarily increase the attack surface of a system and, as a direct consequence, the risk of a successful compromise. This risk can be further increased if - as is often observed with services that are not required - a targeted examination and optimization of the configuration with regard to security does not take place sufficiently.

For this requirement the following threats are relevant:
- Unauthorized use of services or resources
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-5/8.0

Req 2	The accessibility of activated services must be restricted.

In principle, a service provided must be completely deactivated on all interfaces of the system through which accessibility of the service is not required for the proper operation of the system. The deactivation is primarily to be implemented by a corresponding configuration of the service or operating system. In cases where the available configuration options do not allow deactivation on individual interfaces, a local filter ("Host Firewall") may instead be used on the system to block access to the service via unnecessary interfaces.

The accessibility of a service via the required interfaces must also be restricted to legitimate communication partners. The restriction must be implemented by a corresponding configuration of the service or operating system or by means of a local filter ("Host Firewall"). Alternatively, this task may be outsourced to a network-side filter element, provided that the system is located in a separate network segment and communication with this segment is only possible via the network-side filter element.

Motivation: By deactivating services on interfaces through which accessibility is not necessary, as well as by restricting possible communication partners, the attack surface offered by a system can be greatly reduced.

Implementation example: An SNMP service used to monitor a system is enabled exclusively on the dedicated management network interface of the system. A firewall also regulates that only the legitimate monitoring system of the infrastructure environment can reach this service.

For this requirement the following threats are relevant:
- Unauthorized use of services or resources
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-6/8.0

Req 3	Unused software must not be installed or must be uninstalled.

Software could be installed during setup of Linux operating system which is not needed for the functionality of the server. Such software should not be installed or must be uninstalled after installation. Examples for software that is typically not needed on a server system are:

- inetd

- xinetd
- X Window System
- Avahi Server
- CUPS
- rsync service
- NIS server/client
- talk server/client
- telnet server/client
- tftp server/client
- ftp server/client
- rsh server/client

Note: It is not allowed to install software on a server that is not needed for operation, maintenance or general functionality!

Motivation: Vulnerabilities in software offer an attack window for attackers to infiltrate the system. By uninstalling not needed software the attack surface and the risk of a successful compromise can be reduced.

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
- Disruption of availability

ID: 3.65-3/8.0

Req 4	Features that are not required in the software and hardware used must be deactivated.

During the initial installation of software, features may have been activated by default that are not necessary for the operation and functionality of the specific system. Features are usually an integral part of the software that cannot be deleted or uninstalled individually.

Such features must be disabled immediately after the initial installation through the software's configuration settings, so that they remain permanently disabled even after the system is rebooted.

Even before delivery or during initial commissioning, features may have been activated by default in the hardware that are not required for the purpose of the specific system. Such functions, for example unnecessary interfaces, must also be permanently deactivated immediately after initial commissioning.

Motivation: A system's hardware or software often contains enabled features that are not being used. Such features can be an unnecessary target for manipulation. Furthermore, there is a potential that unauthorized access to areas or data of the system can be created.

Implementation example: [Example 1]
Deactivation of debugging functions in the software that are used in the event of fault analysis, but do not have to be active during normal operation.

[Example 2]
Disabling unused network interfaces of a server.

For this requirement the following threats are relevant:
- Unauthorized use of services or resources
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-4/8.0

Req 5	Dedicated partitions must be used for growing content that can influence the availability of the system.

An own partition must be created for directories that are used to store dynamic content. It is recommended to use a dedicated partition for the directories:

- /tmp
- /var

In specific cases it could be necessary to use the following partitions:

- /var/log and /var/tmp (instead of /var)
- /home (for file server with large user base)

Motivation: A filled filesystem can stop operation of a server. This can be triggered by an attacker to effect availability of a server.

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

ID: 3.65-5/8.0

Req 6	Parameters nodev, nosuid and noexec must be set for partitions where this is applicable.

The named mount options must be set for the following partitions if they exist:

- /tmp (nodev, nosuid)
- /var/tmp (nodev, nosuid, noexec)
- /home (nodev)

Note: For installation reasons it could be necessary to remove 'noexec' from partition '/tmp' as this is used sometimes for script execution during software installation.

If separate partition exists also for:

- /dev/shm (RHEL, SLES) (nodev, nosuid, noexec)
- /run/shm (Ubuntu) (nodev, nosuid, noexec)

Motivation: It must be avoided for such partitions that an attacker can execute files with suid, to store device files and to save and execute files from this partition.

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

ID: 3.65-6/8.0

Req 7	Automounting of filesystems using "`autofs`" must be disabled.

Automounting of file systems must be disabled to avoid the automated mounting and use of external file systems like USB sticks and CD-ROMs.

Motivation: With automounting enabled any external file system will be mounted to the server and can possibly misused.

Implementation example: Debian-based Linux:
```
$ sudo apt purge autofs
```
RedHat-based Linux:
```
$ sudo yum remove autofs
```

OR

```
$ sudo systemctl disable --now autofs
```

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

ID: 3.65-7/8.0

Req 8	USB storage must be disabled, this applies to both virtual and physical servers.

Kernel module "usb-storage" serves as driver for USB mass storage device and must be prevented from being loaded by default. To accomplish this "modprobe" must be configured to not load "usb-storage" at system boot.
If need the module can be loaded at runtime, be used and be removed afterwards.

Motivation: USB storage devices are usually not used in productive server operation. Since they can be used to introduce malware or exfiltrate data they must be disabled.

Implementation example: `/etc/modprobe.d/usb-storage.conf` contains:
```
install usb-storage /bin/true
```

Alternatively it can also be used
```
install usb-storage /bin/false
```
instead, if user should become negative feedback when trying to load "usb-storage" module.

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

ID: 3.65-8/8.0

Req 9	The use of "`at`" and "`cron`" must be restricted to authorized users.

The use of the tools 'cron' and 'at', that can be used to schedule automated execution of jobs on a Linux system, must be restricted to authorized users.

Motivation: Users can misuse these tools to execute jobs on a system.

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

ID: 3.65-9/8.0

Req 10	Sticky bit must be set on all world-writable directories.

The sticky bit on a directory means that a file in that directory can be renamed or deleted only by the owner of the file,

by the owner of the directory, and by a privileged process.

Motivation: Setting the sticky bit on world writable directories prevents users from deleting or renaming files in that directory that are not owned by them.

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

ID: 3.65-10/8.0

Req 11	No regular files that are world writable must exist.

World writable files are files that have write permission set for other. These files are writable by any user of the server. Such files must be detected and if existing the rights of these files must be changed to an adequate level.

Motivation: Data in world-writable files can be read, modified, and potentially compromised by any user on the system.

Implementation example: Find world-writeable files in entire filesystem with:

```
sudo find / -type f -perm -002
```

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

ID: 3.65-11/8.0

Req 12	Sessions must be automatically terminated after a period of inactivity adapted to the intended use.

It is necessary that sessions on a system are automatically terminated after a specified period of inactivity.

For this reason, a time-out for sessions must be set. The time period to be selected here depends on the use of the system and, if applicable, the physical environment. For example, the time-out for an application in an unsecured environment must be shorter (a few minutes) than the time-out for an application used by operations personnel for system monitoring tasks in an access-protected area (60 minutes or more).

Motivation: For an open but unused session, there is a risk that an illegitimate user may take over and continue it unnoticed in order to exercise unauthorized access to the system and the data contained therein on behalf of the affected user.

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

ID: 3.01-18/8.0

Req 13	The default user "umask" must be 027 or more restrictive.

The setting of the umask defines which mode files or directories get when they are created by a user. The default umask on most Linux systems is less strict. This is the reason why a stricter umask must be configured.

A umask of 027 is required. This defines the permissions 'read, write, execute' (0) for the user, 'read, execute' (2) for group and no permissions (7) for others.

Motivation: With a strict umask the manipulation of files by unauthorized users can be prevented.

Implementation example: For systems with `bash` and `sh` only `/etc/profile.d/set_umask.sh` contains:

```
umask 027
```

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: 3.65-13/8.0

Req 14	Not needed SUID and SGID bits must be removed from executables.

Executables with SUID or SGID bits set run with extensive rights. Such executables pose a security risk. Therefore, executables with SUID and SGID bit set must be limited to the absolutly needed ones. From all others, the SUID and SGID bits must be removed.

An alternative is to grant more granular permission for such commands with Posix capabilities. This solution allows to enable only needed system functions for a binary file and not full root privileges as with SUID/SGID. It is highly recommended where even possible to use Posix capabilities instead of SUID/SGID!

The following executables are allowed to run with SUID and SGID if not Posix capabilities can be used:

- /bin/ping
- /sbin/pam_timestamp_check
- /sbin/unix_chkpwd
- /usr/bin/at
- /usr/bin/gpasswd
- /usr/bin/locate
- /usr/bin/newgrp
- /usr/bin/passwd
- /usr/bin/ssh-agent
- /usr/libexec/utempter/utempter
- /usr/sbin/lockdev
- /usr/sbin/sendmail.sendmail
- /usr/bin/expiry
- /bin/ping6
- /usr/bin/traceroute6.iputils
- /sbin/mount.nfs
- /sbin/umount.nfs
- /sbin/mount.nfs4
- /sbin/umount.nfs4
- /usr/bin/crontab
- /usr/bin/wall
- /usr/bin/write
- /usr/bin/screen
- /usr/bin/mlocate
- /usr/bin/chage
- /usr/bin/chfn
- /usr/bin/chsh
- /bin/fusermount

- /usr/bin/pkexec
- /usr/bin/sudo
- /usr/bin/sudoedit
- /usr/sbin/postdrop
- /usr/sbin/postqueue
- /usr/sbin/suexec
- /usr/sbin/ccreds_validate
- /usr/lib/dbus-1.0/dbus-daemon-launch-helper
- /usr/lib/policykit-1/polkit-agent-helper-1

Motivation: When an executable has the SUID or SGID bit set, the program runs with the privileges of the file's owner or group instead of the user who is executing the program or its default group. Executables with SUID and SGID pose a risk because vulnerabilies contained could possibly lead to compromise of the system.

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
- Disruption of availability
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.65-14/8.0

Req 15	Core dumps must be disabled and crash reporting softwre must be removed.

A core dump includes complete memory content of a crashed or killed executable program. It can contain data with need of protection like passwords or cryptographic keys. Crash reporting software in some cases overrules core dump configuration (e.g. "Apport", "abrt") and it can automatically send reports to its vendor, thus as the case may be disclose data with need of protection.
Core dumps must be disabled during normal operation and may only be enabled in case of debugging. Errer reporting software must be removed, in case it cannot be removed it must be disabled.
On systems where core dumps are needed it is necessary to disable core dumps for setuid (`fs.suid_dumpable=0`) processes.

Motivation: Core dumps and error reports can include data with need of protection. To avoid information leakage core dumps and crash reporting must be disabled.

Implementation example: 1) uninstall crash reporting software (because could otherwise overridde kernel.core_pattern) :
Debian example:
```
$ sudo apt remove reportbug
```
Ubuntu example:
```
$ sudo apt remove apport
```
Redhat example:
```
$ yum remove abrt
```

2) Find & remove any existing "kernel.core_pattern" setting and then configure it:
```
sudo grep -r 'kernel.core_pattern' /etc/sysctl.*
```

`/etc/sysctl.d/50-coredump.conf` contains:
```
kernel.core_pattern=|/bin/false
```

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

ID: 3.65-15/8.0

Req 16	Protection against buffer overflows must be enabled.

A protection function against buffer overflow attacks must be used on Linux servers. The following Kernel option must be configured:

- kernel.randomize_va_space=2 (seekernel.org, default setting on most Linux distributions)

Precondition is Processor feature

- NoExecute (NX) respectively
- eXecute Disable (XD)

which must be activated in system BIOS. If existing Prelink must be disabled if the ASLR feature PIE (Position-independent executable) is used. This is a tool to modify ELF shared libraries and ELF dynamically linked binaries. In modern Linux distributions like RHEL, SLES Prelink is obsolete and no longer included.
Self-compiled software needs to be compiled with support for these security features. GCC default settings fulfill that.

Motivation: Buffer overflow attacks can be used to unauthorizedly execute code on a system to influence availability or to get full access to a system.

Implementation example: `/etc/sysctl.conf` contains:

```
kernel.randomize_va_space=2
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Disruption of availability

ID: 3.65-16/8.0

2.1. Network Hardening

Req 17	IPv4 protocol stack must be securely configured.

If IPv4 is not used it must be completely disabled. Otherwise, the IPv4 stack on Linux servers must be hardened. For this the following configuration must be implemented:

- IPv4 forwarding must be disabled.
- IPv4 redirects must not be accepted.
- Secure IPv4 redirects must not be accepted.
- IPv4 packet redirect sending must be disabled.
- IPv4 source routed packets must not be accepted.
- Suspicious packets must be logged
- Broadcast ICMP requests must be ignored.
- Bogus ICMP responses must be ignored.
- Reverse Path Filtering must be enabled.
- TCP SYN Cookies must be enabled.
- TCP Timestamps must use random time offset (since Kernel 4.12) or be disabled (prior Kernel 4.12).
- An ICMP ratelimit must be configured.
- ARP must be restricted.

Motivation: An unhardened IPv4 protocol stack is vulnerable against several attacks like denial of service, traffic high jacking.

Implementation example: Example for a Debian-based Distribution with Kernel >= 4.12:

```
vim /etc/sysctl.d/80-ipv4-hardening.conf

net.ipv4.ip_forward=0
net.ipv4.conf.all.accept_redirects=0
net.ipv4.conf.default.accept_redirects=0
net.ipv4.conf.all.secure_redirects=0
net.ipv4.conf.default.secure_redirects=0
net.ipv4.conf.all.shared_media=0
net.ipv4.conf.default.shared_media=0
net.ipv4.conf.all.send_redirects=0
net.ipv4.conf.default.send_redirects=0
net.ipv4.conf.all.accept_source_route=0
net.ipv4.conf.default.accept_source_route=0
net.ipv4.conf.all.log_martians=1
net.ipv4.conf.default.log_martians=1
net.ipv4.icmp_echo_ignore_broadcasts=1
net.ipv4.icmp_ignore_bogus_error_responses=1
net.ipv4.conf.all.rp_filter=1
net.ipv4.conf.default.rp_filter=1
net.ipv4.tcp_syncookies=1
net.ipv4.tcp_timestamps=1
net.ipv4.icmp_ratelimit=100
net.ipv4.icmp_ratemask=88089
net.ipv4.conf.all.arp_ignore=2
net.ipv4.conf.default.arp_ignore=2
net.ipv4.conf.all.arp_announce=2
net.ipv4.conf.default.arp_announce=2
net.ipv4.conf.all.arp_notify=0
net.ipv4.conf.default.arp_notify=0
net.ipv4.conf.all.arp_accept=0
net.ipv4.conf.default.arp_accept=0
```

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

ID: 3.65-17/8.0

Req 18	If IPv6 protocol is used, its protocol stack must be securely configured, otherwise IPv6 must be disabled explicitly.

If IPv6 is not used it must be completely disabled. Otherwise, the IPv6 stack on Linux servers must be hardened. For this the following configuration must be implemented:

- IPv6 forwarding must be disabled.
- IPv6 redirects must not be accepted.
- IPv6 source routed packets must not be accepted.
- IPv6 router advertisements must not be accepted.
- IPv6 router solicitations messages must not be sent.
- IPv6 autoconfiguration must be disabled

Motivation: An un-hardened IPv6 protocol stack is vulnerable against several attacks like denial of service, traffic hijacking.

Implementation example: Example configuration to harden IPv6 for a Debian-based Linux distribution:

```
/etc/sysctl.d/80-ipv6-hardening.conf

net.ipv6.conf.all.forwarding=0
net.ipv6.conf.all.accept_redirects=0
net.ipv6.conf.all.accept_source_route=0
net.ipv6.conf.all.accept_ra=0
net.ipv6.conf.all.router_solicitations=0
net.ipv6.conf.all.autoconf=0
net.ipv6.conf.default.forwarding=0
net.ipv6.conf.default.accept_redirects=0
net.ipv6.conf.default.accept_source_route=0
net.ipv6.conf.default.accept_ra=0
net.ipv6.conf.default.router_solicitations=0
net.ipv6.conf.default.autoconf=0
```

Example configuration to disable IPv6 for a Debian-based Linux distribution:

```
/etc/sysctl.d/80-ipv6-disable.conf

net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
```

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

ID: 3.65-18/8.0

2.2. System Update

Req 19	The software used must be obtained from trusted sources and checked for integrity.

The software used on the system must be obtained from trusted sources and checked for integrity before installation. This requirement applies to all types of software:

- Firmware and microcode for hardware components
- Operating systems
- Software Libraries
- Application Software
- Pre-integrated application solutions, such as software appliances or containers

as well as other software that may be used.

Trusted Sources
Trusted sources are generally considered to be:

- the official distribution and supply channels of the supplier
- third party distributors, provided they are authorized by the supplier and are a legitimate part of the supplier´s delivery channels
- internet downloads, if they are made from official provisioning servers of the supplier or authorized distributors
  (1) If the provisioning server offers various forms of downloads, those protected by encryption or cryptographic signatures must be preferred to those without such protection.
  (2) If the provisioning server secures the transport layer using cryptographic protocols (e.g. HTTPS, sftp), the associated server certificates or server keys/fingerprints must be validated with each download to confirm the identity of the provisioning server; if validation fails, the download must be cancelled and the provisioning server has to be considered an untrusted source.

Integrity Check

The integrity check is intended to ensure that the received software is free of manipulation and malware infection. If available, the mechanisms implemented by the supplier must be used for checking.
Valid mechanisms are:

- physical seals or permanently applied certificates of authenticity (if the software is provided on physical media)
- comparison of cryptographic hash values (e.g. SHA256, SHA512) of the received software against target values, which the supplier provides separately
- verification of cryptographic signatures (e.g. GPG, certificates) with which the supplier provides its software

In addition, a check of the software using an anti-virus or anti-malware scanner is recommended (if the vendor has not implemented any of the aforementioned integrity protection mechanisms for its software, this verification is mandatory).

Extended integrity checking when pulling software from public registries

Public registries allow developers to make any of their own software projects available for use. The range includes projects from well-known companies with controlled development processes, as well as from smaller providers or amateur developers.
Examples of such registries are:

- Code registries (e.g. GitHub, Bitbucket, SourceForge, Python Package Index)
- Container registries (e.g. Docker Hub)

Software from public registries must undergo an extended integrity check before deployment.
In addition to the integrity check components described in the previous section, the extended check is intended to explicitly ensure that the software actually performs its function as described, does not contain inherent security risks such as intentionally implemented malware features, and is not affected by known security vulnerabilities. If the software has direct dependencies on third-party software projects (dependencies are very typical in open source software), which must also be obtained and installed for the use of the software, these must be included in the extended integrity check.

Methods for an extended integrity check can be, for example:

- Strict validation of project/package names (avoidance of confusion with deliberately imitated malicious software projects)
- dynamic code analysis / structured functional checks in a test environment
- static code analysis using a linter (e.g. Splint, JSLint, pylint)
- Examination using a security vulnerability scanner (e.g. Qualys, Nessus)
- Examination using a container security scanner (e.g. JFrog Xray, Harbor, Clair, Docker Scan)
- Examination using an SCA (Software Composition Analysis) tool or dependency scanner (e.g. OWASP Dependency Check, Snyk)

The test methods must be selected and appropriately combined according to the exact form of software delivery (source code, binaries/artifacts, containers).

Motivation: Software supply chains contain various attack vectors. An attacker can start at various points to manipulate software or introduce his own routines and damage or control the target environment in which the software is subsequently used. The attack can occur on the transport or transmission path or on the provisioning source itself. Accordingly, an attack is facilitated if software is not obtained from official and controlled sources or if an integrity check is omitted.
There is a particular risk for software obtained from public registries, as these are open to anyone for the provision of software projects. Perfidious attack methods are known, in which the attacker first provides completely inconspicuous, functional software for a while and as soon as it has established itself and found a certain spread, deliberately hidden malicious code is integrated in future versions. Other methods rely on similar-sounding project names for widely used existing projects or overruling version numbers to inject manipulated software into any solutions based on them.

Implementation example: Obtain the software via the official delivery channels of the supplier. Upon receipt of the software, immediately check for integrity using cryptographic checksums, as provided by the supplier, as well as scan for any infections by known malware using anti-malware / anti-virus scanners. Storage of the tested software on an internal, protected file storage and further use (e.g. rollout to the target systems) only from there.

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

ID: 3.01-2/8.0

Req 20	Linux distributions which are supported by their vendor must be used. Absolutely necessary is security vulnerability support.

Releases of Linux distributions with active support by their creator (distributor, producer or developer) must be used. Releases which are marked as end-of-support (EOS), end-of-life (EOS) or something the like by their vendors must **not** be used. If available long term support (LTS) versions are to prefer.

Such support must include that the supplier
- continuously monitors and analyzes the product for whether it has been affected by security vulnerabilities,
- informs immediately about the type, severity and exploitability of vulnerabilities discovered in the product
- timely provides product updates or effective workarounds to remedy the vulnerabilities.

The security vulnerability support must be given for the entire period in which the affected product remains in use.

Support phases with limited scope of services
Many suppliers optionally offer time-extended support for their products, which goes beyond the support phase intended for the general market, but is often associated with restrictions. Some suppliers generally define their support in levels that can contain restrictions already in the last phase immediately before the absolute end date of regular support.
If a product is used within support phases that are subject to restrictions, it must be explicitly ensured that these restrictions do not affect the availability of security vulnerability support.

Open Source Software and Hardware
Open Source products are often developed by free organizations or communities; accordingly, contractually agreed security vulnerability support may not be available. In principle, it must also be ensured here that the organization/community (or a third party officially commissioned by it) operates a comprehensive security vulnerability management for the respective product that meets the above criteria and is considered to be reliably established.

Motivation: Safe production can only be achieved with software that obtains timely security patches. Known vulnerabilties and Proof-of-Concepts to exploit them make systems without vendor support an easy target for attackers.

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

ID: 3.65-20/8.0

Req 21	If needed, active software licenses must be installed to ensure security updates.

Motivation: Some operating system vendors license their products and require the purchase of licenses. These allow access to (security) updates. Without update options, a system may not be operated.

Implementation example: Red Hat Enterprise Linux for example can be registered with: `subscription-manager register`

ID: 3.65-21/8.0

Req 22	Known vulnerabilities in the software or hardware of the system must be fixed or protected against misuse.

Known vulnerabilities in software and hardware components must be fixed by installing available system updates from the supplier (e.g. patches, updates/upgrades). Alternatively, the use of workarounds (acute solutions that do not fix the vulnerability, but effectively prevent exploitation) is permissible. Workarounds should only be used temporarily and should be replaced by a regular system update as soon as possible in order to completely close the vulnerabilities.

Components that contain known, unrecoverable vulnerabilities must not be used in a system.

The treatment of newly discovered vulnerabilities must also be continuously ensured for the entire deployment phase of the system and implemented in the continuous operating processes of security patch management.

Motivation: The use of components without fixing contained vulnerabilities significantly increases the risk of a successful compromise. The attacker is additionally favored by the fact that, as a rule, not only detailed information on vulnerabilities that have already become known is openly available, but often also already adapted attack tools that facilitate active exploitation.

Implementation example: Following the initial installation of an operating system from an official installation medium, all currently available patches and security updates are installed.

Additional information:
The primary sources of known vulnerabilities in software/hardware are lists in the release notes as well as the security advisories from the official reporting channels of the supplier or independent CERTs. In particular, the reporting channels are sensibly integrated into continuous processes of security patch management for a system, so that newly discovered vulnerabilities can be registered promptly and led into operational remedial measures.
As a complementary measure to the detection of potentially still contained types of vulnerabilities that have in principle already become known, targeted vulnerability investigations of the system can be carried out. Particularly specialized tools such as automated vulnerability scanners are suitable for this purpose. Examples include: Tenable Nessus, Qualys Scanner Appliance.

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
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-10/8.0

Req 23	GPG check for repository server must be activated and corresponding keys for trustable repositories must be configured.

GPG check must be enabled and keys must be configured properly to verify integrity during installation of software from a repository server. On RedHat Linux it is necessary to activate the gpgcheck globally.

Motivation: The GPG check is necessary to guarantee the authenticity of used source an integrity of software. Without this check an attacker could possibly manipulate software packets before installation.

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

- Disruption of availability

For this requirement the following warranty objectives are relevant:
- Integrity

ID: 3.65-23/8.0

Req 24	The operating system must have an Endpoint Detection and Response (EDR) solution.

An up-to-date Endpoint Detection and Response (EDR) solution must be used on the operating system. An EDR solution collects security-relevant activity data from processes and evaluates them centrally. Alarms can be generated from malicious behavior of processes or by a specific signature (like classic virus scanners). In addition, it is possible to react directly to suspicious program behavior via a central console. For example, the operating system or rather the client can be isolated or the malicious process terminated. Furthermore, an overview of the vulnerabilities of all operatings systems is forwarded to a central location.

Motivation: Normal virus scanners rely purely on signatures, whereas an EDR solution also searches for anomalies in the process behavior. Signatures have the disadvantage that new signatures have to be written for new virus variants before they can be detected.

Implementation example: Have a look at PSA Wiki - Sicherheitsanforderungen : Endpoint Detection and Response (EDR) for guidance.

For this requirement the following threats are relevant:
- Unnoticeable feasible attacks

ID: 3.65-24/8.0

2.3. User Authentication

Req 25	User accounts must ensure the unique identification of the user.

Users must be identified unambiguously by the system.

This can typically be reached by using a unique user account per user.

So-called group accounts, which are characterized by the fact that they are used jointly by several people, must not be used. This also applies without restriction to privileged user accounts. Most systems initially have only a single user account with administrative privileges after the basic installation. If the system is to be administered by several persons, each of these persons must use a personal, individual user account to which appropriate administrative authorizations or roles are assigned

A special feature are so named technical user accounts. These are used for the authentication and authorization of systems among themselves or of applications on a system and can therefore not be assigned to a specific person. Such user accounts must be assigned on a per system or per application basis. In this connection, it has to be ensured that these user accounts can't be misused.
Ways to prevent misuse of such user accounts by individuals include:

- Configuration of a password that meets the security requirements and is known to as few administrators as possible.
- Configuring the user account that only a local use is possible and a interactive login isn't possible.
- Use of a technique for authentication of the specific user account with public and private key or certificates.
- Limiting the access over the network to legitimate systems.

Additional solution must be checked on their usability per individual case.

Motivation: Unambiguous user identification is mandatory to assign a user permissions that are necessary to perform

the required tasks on the system. This is the only way to adequately control access to system data and services and to prevent misuse. Furthermore, it makes it possible to log activities and actions on a system and to assign them to individual users.

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

ID: 3.01-22/8.0

Req 26	System accounts must be non-login.

On Linux servers, several users are available that are needed for functionality of applications. These users are not intended to provide a shell. To avoid that such accounts are used to login their shell must be disabled.

Note: The system accounts root, sync, shutdown and halt are excluded from this requirement!

Motivation: This prevents system accounts from potentially being misused by attackers to run commands.

Implementation example: To prevent such users from logging in shell in "`/etc/passwd`" can be set to "`/usr/sbin/nologin`" or "`/bin/false`".

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

ID: 3.65-26/8.0

Req 27	User accounts must be protected with at least one authentication attribute.

All user accounts in a system must be protected against unauthorized use.

For this purpose, the user account must be secured with an authentication attribute that enables the accessing user to be unambiguously authenticated. Common authentication attributes are e.g.:

- passwords, passphrases, PINs (factor KNOWLEDGE: "something that only the legitimate user knows")
- cryptographic keys, tokens, smart cards, OTP (factor OWNERSHIP: "something that only the legitimate user has")
- biometric features such as fingerprints or hand geometry (factor INHERENCE: "something that only the legitimate user is")

The authentication of users by means of an authentication attribute that can be faked or spoofed by an attacker (e.g. telephone numbers, IP addresses, VPN affiliation) is generally not permitted.

In companies of Deutsche Telekom group where the MyCard or a comparable smartcard is available this should be a preferred authentication attribute.

If the system and the application scenario support it, multiple independent authentication attributes should be combined if possible in order to achieve an additional increase in security (so-called MFA or Multi-Factor-Authentication).

Motivation: User accounts that are not protected by appropriate authentication attributes can be abused by an attacker to gain unauthorized access to a system and the data and applications stored on it.

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

ID: 3.01-20/8.0

Req 28	Privileged user accounts must be protected with at least two authentication attributes from different factors.

A privileged user account is a user account with extended authorizations within a system. Extended authorizations enable access to configuration settings, functions or data that are not available to regular users of the system. In direct dependence on the special tasks that are carried out via a privileged user account within a system, the assigned extended authorizations can be specifically restricted or include completely unrestricted system access.

Examples of privileged user accounts:

- Accounts for administration, maintenance or troubleshooting tasks
- Accounts for user administration tasks (e.g. creating/deleting users; assigning permissions or roles; resetting passwords)
- Accounts that are authorized to legitimize, initiate or prevent business-critical processes
- Accounts that have access to data classified as SCD (Sensitive Customer Data) in the interests of Group Deutsche Telekom, its customers or the public
- Accounts that have extensive access to data defined as "personal" according to the EU-GDPR (e.g. mass retrieval of larger parts or the complete database)

A single authentication attribute for privileged user accounts with their extended authorizations is usually no longer sufficient.

In order to achieve an adequate level of protection, at least two mutually independent authentication attributes must be used. The authentication attributes must come from various factors (knowledge, ownership, inherence). A combination of authentication attributes of the same factor (e.g. two different passwords) is not permitted

This approach is commonly referred to as MFA (Multi-Factor Authentication).
A specific form of MFA is 2FA (2-factor authentication), which combines exactly two authentication attributes.

Motivation: Privileged user accounts represent an increased risk to the security of a system. If an attacker successfully compromises such a user account, he receives extensive authorizations with which he can bring the system or system parts under his control, disrupt system functions, view/manipulate processed data or influence business-critical processes. The combination of multiple authentication attributes of different types significantly minimizes the risk of a user account being compromised.

Implementation example: Very popular is 2FA in a variant consisting of an attribute that the user knows (factor KNOWLEDGE) and an attribute that the user possesses (factor OWNERSHIP).
Examples of such a 2FA are:

- smartcard (e.g. MyCard) plus PIN
- private key plus passphrase
- classic password plus hardware token for the generation of OTPs

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

ID: 3.01-21/8.0

Req 29	Authentication must be enabled for "single-user mode" and bootloader (e.g. GRUB 2). Also number of local accounts needed for operation must be minimized.

For system recovery, the so called "single-user mode" is used. This mode can also be manually selected from the bootloader during system boot. Authentication must be enabled for single user mode to protect this access. This is especially relevant, if the console of the machine can be reached remotely (e.g. via lights-out-management or via virtual console in case of virtual machines).

Further it must be ensured that all non-removeable unused local accounts are disabled. Accounts where a login in certain cases must be possible with a password (root, grubadmin), must have passwords for each machine and each account individually and with no apparent formation rule.

Motivation: Local accounts are additional points of attack which can be used by attackers or unauthorized individuals. This requires that only absolutely necessary local accounts required for operation exist in the operating system. Without authentication, an unauthorized user can gain root privileges.

Implementation example: Debian-based Distro example:
1) Set a password for single-user mode, which is root password ultimately: `$ sudo passwd root`
2) Configure GRUB 2 bootloader.
2.1) Generate a sufficiently iterated password hash: `$ grub-mkpasswd-pbkdf2 -c 640000`
2.2) Put user and hash into GRUB 2 config:
`/etc/grub.d/40_custom` contains:

```
set superusers="grubadmin"
password_pbkdf2 grubadmin grub.pbkdf2.sha512.640000.6B4ABEE1920FDE...
```

2.3) Configure boot without password entry:
`/etc/grub.d/10_linux` contains variable "`CLASS`" with parameter "`--unrestricted`":

```
CLASS="--class gnu-linux --class gnu --class os --unrestricted"
```

2.4) Update GRUB 2 config: `$ sudo update-grub`

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

ID: 3.65-29/8.0

2.4. Administrative Access

Req 30	The administration of the operating system must be done via a network interface which is independent from the production network.

Administrative access to a server must not be done via an interface which provides productive services. Access must be limited to legitimate systems. The administration of applications can also be done using this network interface.

The restriction can be done with, e.g., filter mechanisms, local access lists or a packet filter. This limitation has to be done as restrictive as possible, i.e., limit to single IP addresses or at least small IP ranges.

Motivation: In the event of a successful attack, an attacker may gain access to confidential information or even to the entire system. By restricting the accessibility to legitimate systems, the group of potential attackers can be reduced, and thus also the likeliness of a successful attack. Furthermore, systems must be manageable even in the case the customer network is down.

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

ID: 3.37-10/8.0

Req 31	Administrative services and accesses must be bound to only those interfaces that have been set up to administer.

The administrative services (interactive by persons or machine-to-machine, e.g., SSH, HTTPS, RDP) must be bound to the appropriate interface(s). Due to the separation of management traffic from user traffic, this is the IP address in the management network. If the system - or parts of it - is managed by more than one interface, the management services have to be bound to the lowest possible number.

Depending on the respective service, the access to these services must be restricted to a few, trustworthy, necessary target or source addresses.

Motivation: This ensures that it can be clearly foreseen under which address these management services are reachable. In addition, a unique address is important for implementing filters and firewall rules that restrict access to these management services to legitimate addresses.

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

ID: 3.37-11/8.0

Req 32	Network based access used for operating system administration must have integrity protection, be encrypted and securely authenticated.

Access is only permitted by using secure protocols (e.g., SSHv2, HTTPS, SNMPv3). The administrator must ensure that any network connection between his workstation or a management system and the operating system to be administrated is securely authenticated, encrypted and protected against tampering.

Motivation: If the administrator transmits changes to the configuration settings via unencrypted or unsecure connections, there is a risk that unauthorized parties gather system information (configuration settings, access IDs, etc.) to exploit security vulnerabilities.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Denial of executed activities

ID: 3.37-12/8.0

2.5. Location

Req 33	If the system is not located in a room with at least protection class "high" (PC3), the BIOS and, if available, other options for local management must be secured against unauthorized access.

The Protection Classes (PC) are defined in the Annex 1, "Physical Security of Buildings", of the Group Policy on "Physical Security". Typically, Datacenters are compliant to the requirements of PC3.

Servers operated in public or customer areas must be especially protected against unauthorized access and changes: The BIOS settings must be protected against export and tampering. Are further access options to the system configuration possible, e.g. by Intel AMT, iLO, LOM, and others, these must be protected as well. In case passwords are used, these must be exclusive to the individual server and must not allow conclusions to be drawn about a distinguishing feature of the server.

The BIOS must be configured in such a way that only the designated operating system can be started with it from the designated partition.

Motivation: Motivation: Changing BIOS settings can facilitate attacks. Since, for example, local rooms with technical installations seldom offer access protection to the servers, attackers could change the startup sequence of data storage media when the server is started in the BIOS without the password protection described. This would make it possible to start an alternative operating system which circumvents the security mechanisms of the implemented operating system.

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

ID: 3.37-20/8.0

Req 34	If the system is not located in a room with at least protection class "high" (PC3), used data storages must be fully encrypted.

The Protection Classes (PC) are defined in the Annex 1, "Physical Security of Buildings", of the Group Policy on "Physical Security". Typically, Datacenters are compliant to the requirements of PC3.

Data storages are all disks and flash memory in the systems.

Motivation: Access to devices which are operated outside of data centers with protected access is relatively easy. Physical data storage media can be easily stolen as a result.

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

ID: 3.65-34/8.0

3. Logging

Req 35	Auditing must be enabled at boot by setting a kernel parameter.

Each process on the system carries an "auditable" flag which indicates whether its activities can be audited. Although "auditd" takes care of enabling this for all processes which launch after it does, adding the kernel argument ensures it is set for every process during boot.

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 against attackers.

Implementation example: For Debian based distributions e.g. you can edit `/etc/default/grub` . Go for variable `GRUB_CMDLINE_LINUX_DEFAULT` and add "`audit=1`" for the kernel parameter to activate. After doing this change it is need to issue `update-grub` of course for the changes to take effect.

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

ID: 3.65-35/8.0

Req 36	Log rotation for logfiles must be configured.

Log rotation for logfiles must be enabled. Files must be rotated based on file size and max lifetime of file.

Motivation: An attacker can trigger log events to fill up the disk space of the server. This could lead to a denial of service of the server.

Implementation example: logrotate configuration contains:# restrict maximum size with maxsize and rotate

```
# restrict maximum size with maxsize and rotate
maxsize 10M
rotate 100

# control maximum age with daily rotation and maxage
daily
maxage 90

# permissions
create 640 syslog adm

# secure deletion
shred
```

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

ID: 3.65-36/8.0

Req 37	System time must be synchronized against a reference time source.

A service like NTP (client programs: ntp, crony) must be used for time synchronization. Example for Deutsche Telekom AG NTP server pools are:

- ntp-pool-info_ntpp10.telekom.de
- ntp-pool-info_ntpp21.telekom.de

Motivation: Time synchronization must be done on a server to support time sensitive security mechanisms like the usage of certificates or solutions like Kerberos. A precise time is also mandatory to ensures consistent time records for logging events.

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

ID: 3.65-37/8.0

Req 38	Applicable retention and deletion periods must be observed for security-relevant logging data that is recorded locally.

From an IT security perspective, local storage of security-relevant logging data on a system is not mandatory. Since the local storage can be damaged in the event of system malfunctions or manipulated by a successful attacker, it can only be used to a limited extent for security-related or forensic analyses. Accordingly, it is relevant for IT security that logging data is forwarded to a separate log server.

Local storage can nevertheless take place; for example, if local storage is initially indispensable when generating the logging data due to technical processes or if there are justified operational interests in also keeping logging data available locally.

The following basic rules must be taken into account when storing logging data locally:
- Security-related logging data must be retained for a period of 90 days.
  (*This requirement only applies if no additional forwarding to a separate log server is implemented on the system and the logging data is therefore only recorded locally.*)
- After 90 days, stored logging data must be deleted immediately.

Deviances
Different retention periods and deletion periods may exist due to legal or regulatory requirements (especially in connection with personal data) or may be defined by contractual agreements. In these cases, the applicable periods must be agreed individually with a Project Security Manager (PSM) / Data Privacy Advisor (DPA) or are specified by them.

Motivation: Logging data is an immensely important IT security tool for preventing, detecting and clearing up system faults, security and data privacy incidents. On the other hand, the recording of logging data, like any other data processing, is also subject to legal and regulatory requirements. Accordingly, guidelines must be adhered to that reconcile the two.

Implementation example: Using logrotate old logs can be deleted after 90 days. Option "shred" makes sure data is deleted safely (at host level). As a rule, the following example cannot be adopted 1:1 in addition to distro standards. The options "`daily`" & "`maxage`" as well as "`shred`" are primarily intended to illustrate how an implementation can be realized.

```
$ cat /etc/logrotate.d/any-log
/var/log/* {
  daily
  maxage 90
  shred
}
```

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

ID: 3.65-38/8.0

Req 39	Security-relevant logging data must be provided to a security incident detection and response solution in near-real-time.

An acceptable detection and response solution for the purposes of this requirement must meet the following criteria:

- There is a continuous analysis and correlation of the logging data provided by connected IT/NT systems.
- The identification of threatening activities and ongoing attacks takes place on the basis of a continuously updated detection pattern database
- Integrated alerting processes trigger the initiation of countermeasures in the event of a find.

Such a solution is not the same as a log server in terms of its function. Accordingly, the provision to a detection and response solution must always be in addition to an existing forwarding to a log server. **An exception exists if the log server used provides the logging data to a detection and response solution immediately after receipt.** *In this case, it is sufficient for the system to forward the logging data only to the log server.*
The logging data must be provided to the detection and response solution within 5 minutes ("near-real-time") from the time it is generated in the system.

The specific technical interfaces and protocols to be supported for deployment are determined by the detection and response solution available in the system's operating environment as well as the technical basis of the system itself.

Motivation: IT/NT systems are constantly exposed to security threats and attacks. By means of a central analysis and the correlation of recorded system events of entire IT landscapes in almost real time, current threats and ongoing attacks can be detected even beyond system boundaries and appropriate countermeasures can be initiated promptly. The prompt delivery of logging data to the detection and response solution is essential for its effectiveness.

Implementation example: **Example 1:** A system based on a classic server collects security-relevant logging data in the operating system's local syslog. Logging data is continuously forwarded to a Security Information and Event Management (SIEM) for further analysis, intrusion detection, and alarm handling.

Example 2: An Endpoint Detection and Response (EDR) agent is running on a system. The EDR agent continuously forwards the security-related logging data it collects on the system to the central backend of the EDR solution, where further analysis, intrusion detection, and alarm handling takes place

ID: 3.65-40/8.0

3.1. Auditd

The auditd subsystem is an access monitoring and accounting for Linux. It can be used to define granular log events to be monitored under Linux OS. From security point of view log events can be used to detect malicious activities and to analyze possible breaches.

Req 40	"auditd" service must be used to log security relevant events.

On Linux server, "auditd" must be installed and configured to log security relevant events. Every event must be logged with a precise timestamp and a unique system reference.
Logdata of "auditd" counts as security reelvant event that has to be forwarded to a central log server, as described in chapter "External Logging".

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 against attackers.

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

ID: 3.65-41/8.0

Req 41	Syscalls "execve" (execute program) must be logged.

On Linux servers, the following System events must be logged:

Event Type	Description	Category
execution of programs	All program executions must be logged.	Mandatory

Motivation: Forensics & mitigation *: In case of compromise of a Linux host it needs to be comprehensible which actions have been taken.*

Implementation example: As example an Ubuntu 20.04 host could be configured like this:

```
$ cat /etc/audit/rules.d/execve.rules
-a exit,always -F arch=b64 -S execve
-a exit,always -F arch=b32 -S execve
```

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

ID: 3.65-42/8.0

Req 42	System events must be logged.

On Linux servers, the following System events must be logged:

Event Type	Description	Category
System Startup and Shutdown	All restarts or shutdowns of the OS must be logged.	Mandatory
(Un)Installation of software	After the commissioning of the server, every uninstallation and installation of software must be logged.	Mandatory
Change of system time	Modification of the local system time and change of ntp settings must be logged.	Mandatory
Connection of external device (storage)	The connection of external devices like USB-Flash drives, which can mount on the running server, must be logged.	Mandatory
Privileged commands execution	The use of privileged commands with SUID/SGID must be logged.	Mandatory
Loading/unloading of kernel modules	The loading and unloading of kernel module must be logged.	Mandatory
Change of scheduled jobs	Jobs which are executed periodically, must be monitored, if they are changed or deleted.	Optional

Motivation: It is unusual to make system changes when the OS is put into operation. An attacker who has access to the server could change the system for its malicious purpose. The logging of the system events is necessary to detect and backtrack these attacks.

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

ID: 3.65-43/8.0

Req 43	Access and Authentication events must be logged.

On Linux servers, the following Access and Authentication events must be logged:

Event Type	Description	Category
Logon and Logoff	The Logon and Logoff of a User via external or local access must be logged.	Mandatory
Password Change	User Password changes or resets must be logged.	Mandatory
Escalation of privileges (sudo/sudoers)	It must be logged, if a user executes sudo or changes corresponding configuration files (sudoers).	Mandatory
Modification of DAC permissions	Modification of discretionary access control permissions must be logged.	Mandatory

Motivation: The logging of authentication and access events can be useful to backtrack who has access to a certain time. With these logs it is for example possible to detect a captured account which is used by an attacker.

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

ID: 3.65-44/8.0

Req 44	Account and Group Management events must be logged.

On Linux servers, the following Account and Group Management events must be logged:

Event Type	Description	Category
Creation, modification and deletion of users	It must be logged, if users are created, modified or deleted.	Mandatory
Creation, modification and deletion of groups	It must be logged, if a groups are created, modified or deleted.	Mandatory

Motivation: The logging of account and group management events can be useful to backtrack user and group management. With these logs it is possible to detect malicious modification, creation and deletion of users and groups.

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

ID: 3.65-45/8.0

Req 45	Configuration Change events must be logged.

On Linux servers, the following Configuration Change events must be logged:

Event Type	Description	Category
Disable logging	It must be logged, if the logging service is disabled.	Mandatory
Deletion and unauthorized modification of logs	The deletion of events must be logged. The unauthorized modification of logs must be logged.	Mandatory
Change of logging configuration	It must be logged, if there is a change of the logging configuration	Mandatory
Change of network configuration	Change of network and interface configuration must be logged.	Mandatory
Authentication Subsystem changes	Changes of Authentication Subsystems (e.g. LDAP- or Kerberos-Policy) must be logged.	Optional
Critical File changes	Depending on the use case, critical file changes should be logged.	Optional
Change of SELinux configuration	If used SELinux events and changes of configuration must be logged.	Optional
Change of AppArmor configuration	If used AppArmorevents and changes of configuration must be logged.	Optional

Motivation: Configuration changes could have a massive impact of the OS and consequently could be a security risk. It is necessary to identify all important configurations on the OS and log the changes.

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

ID: 3.65-46/8.0

Req 46	Auditd configuration must be immutable.

Immutable mode must be set for auditd to avoid that audit rules can be modified with 'auditctl' command.

Motivation: If auditd is not in immutable mode, unauthorized users can initiate changes to hide malicious activity.

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

ID: 3.65-47/8.0

3.2. External Logging

Req 47	Security relevant logging data must be sent to a remote system directly after their creation.

Security relevant logging data must be forwarded to an external system in appropriate logging files as well as being stored locally. Standard protocols like Syslog, SNMPv3 must be preferred. The transfer should be secured, i.e. encrypted and authenticated, when data with need of protection is transmitted. To enable testing for consistency and completeness, sequence numbers are to be used and, if feasible, TCP instead of UDP.

Hint: The receiver of the data must be informed, how the data provided should be assessed.

Motivation: If logging data is only stored locally, it can be manipulated by an attacker who succeeds in compromising the system in order to conceal his attack and any manipulation he has performed on the system. This is the reason why the forwarding must be done immediately after the event occurred.

Implementation example: rsyslog configuration contains:

```
*.* @@host1.central-log.internal
*.* @@host2.central-log.internal
```

ID: 3.65-48/8.0

Req 48	For security-relevant logging data that is forwarded to the separate log server, compliance with the applicable retention and deletion periods must be ensured.

The following basic rules must be taken into account:

- security-related logging data must be retained for a period of 90 days on the separate log server.
- after 90 days, stored logging data must be deleted immediately on the separate log server.

Deviances

Different retention periods and deletion periods may exist due to legal or regulatory requirements (especially in connection with personal data) or may be defined by contractual agreements. In these cases, the applicable periods must be agreed individually with a Project Security Manager (PSM) / Data Privacy Advisor (DSB) or are specified by them.

Log server under the responsibility of a third party

If the selected separate log server is not within the same operational responsibility as the source system of the loggin data, it must be ensured that the responsible operator of the log server is aware of the valid parameters for the logging data to be received and that they are adhered to in accordance with the regulations mentioned here.

Motivation: Logging data is an immensely important IT security tool for preventing, detecting and clearing up system faults, security and data privacy incidents. On the other hand, the recording of logging data, like any other data processing, is also subject to legal and regulatory requirements. Accordingly, guidelines must be adhered to that reconcile the two.

Implementation example: Taking into account the current legal situation and applicable data privacy regulations, the following deletion periods for forwarded security-relevant logging data from an exemplary telecommunications system are implemented on the separate log server:

- Standard System Logs: Deletion after 90 days at the latest
- Logging of public IP addresses: Deletion (or anonymization) after 7 days at the latest
- Logging of the assignment of dynamic public IP addresses by the telecommunication solution: Deletion after 7 days at the latest
- Logging of non-billing-relevant call detail records: Deletion after 7 days at the latest
- Logging of the content of e-mail and SMS: Deletion after 24 hours at the latest
- Logging of the domain queries handled by the DNS server of the telecommunications solution: Deletion after 24 hours at the latest

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.01-36/8.0

3.2.1. RSyslog

Req 49	If RSyslog is used, the default permission of 640 or more restrictive for logfiles must be configured.

For RSyslog the default permissions of 640 or more restrictive used for log files must be configured in file 'rsyslog.conf'.

Motivation: Unauthorized access to log files is possible if they have wrong file permissions. This is a risk of unwanted information leakage, as such files can contain data with need of protection.

Implementation example: rsyslog configuration contains:

```
FileOwner syslog
FileGroup adm
FileCreateMode 0640
DirCreateMode 0750
Umask 0027
PrivDropToUser syslog
PrivDropToGroup adm
```

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

ID: 3.65-50/8.0

Req 50	If RSyslog is used, at least one central logging server must be configured.

One or several external log servers must be configured in RSyslog configuration.

Motivation: If logging data is only stored locally, it can be manipulated by an attacker to conceal the attack and any manipulation done on the system. This is the reason why the events must be forwarded immediately after occurrence.

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

ID: 3.65-51/8.0

3.2.2. Syslog-NG

Req 51	If Syslog-NG is used, the default permission of 640 or more restrictive for logfiles must be configured.

For Syslog-NG the default permissions of 640 or more restrictive used for new generated logging files must be configured in file 'syslog-ng.conf'.

Motivation: Unauthorized access to logging files is possible if they have wrong file permissions. This is a risk of unwanted information leakage, as such files can contain data with need of protection.

Implementation example: syslog-ng configuration contains:

```
create-dirs yes
dir-owner syslog
dir-group adm
dir-perm 0750
owner syslog
group adm
perm 0640
```

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

ID: 3.65-52/8.0

Req 52	If Syslog-NG is used, at least one central logging server must be configured.

One or several external log servers must be configured in RSyslog configuration.

Motivation: If logging data is only stored locally, it can be manipulated by an attacker to conceal the attack and any manipulation done on the system. This is the reason why the events must be forwarded immediately after occurrence.

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

ID: 3.65-53/8.0

4. Authentication attribute "password" & Pluggable Authentication Modules

PAM (Pluggable Authentication Modules) is a solution to provide independent modules for authentication schemes to programs. PAM is used by default in most common Linux distributions.

Req 53	If passwords are used as an authentication attribute, "Shadow Password Suite" must be configured to protect passwords by using a secure hashing function. Passwords must be kept at least 1 day and must be changed latest after 365 days.

Passwords must always be stored as hashes. **sha512** with 640.000 rounds and Salt (96 Bit) or **yescrypt** (*shadow* >= `4.14.0`) must be used as a hashing algorithm to protect passwords. **yescrypt** is preferable if it is available. Passwords must be used at least 1 day and up to 365 days. Background of this requirement is described in Security requirement 3.01 Technical Baseline Security for IT/NT Systems Chapter 8. Authentication parameter password.

Motivation: It an unauthorized person gets access to a password file, the password can be misused if not stored in a secure way. Consecutively changing password can be a misused to circumvent protection by password history, therefore a minimum password age is necessary.

Implementation example: z.B. Ubuntu 22.04 LTS "jammy" contains in `/etc/login.defs` :

```
ENCRYPT_METHOD YESCRYPT
PASS_MAX_DAYS 365
PASS_MIN_DAYS 1
```

z.B. Ubuntu 20.04 LTS "focal" contains in `/etc/login.defs` :

```
ENCRYPT_METHOD SHA512
SHA_CRYPT_MIN_ROUNDS 640000
PASS_MAX_DAYS 365
PASS_MIN_DAYS 1
```

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

ID: 3.65-54/8.0

Req 54	If passwords are used as an authentication attribute, "PAM" must be configured to use strong salted password hash functions while doing sufficient calculation rounds to protect passwords.

PAM can be configured to take advantage of the secure cryptographic hash function with salt for passwords. To further increase security, several rounds of the hash function are executed. The default setting for SHA-512 is 5000 rounds. This is not sufficient due to cheap and readily available compute ressources, which make brute force attacks conceivable. Therefore, the number of rounds must be increased to at least 640,000.

Newer cryptographic libraries offer **yescrypt** hash function, which is clearly superior to SHA-512. It is offered for several distributions and is now used as standard. **yescrypt** is preferable if it is available.

Motivation: If an unauthorized person gets access to a password file, the password can be misused if not stored in a secure way.

Implementation example: **Debian-based Distro example:**

- `/etc/pam.d/common-password` needs to contain `yescrypt` option configured : `password [success=1 default=ignore] pam_unix.so obscure yescrypt`
- `/etc/pam.d/common-password` needs to contain `sha512` and `rounds=640000` options configured : `password [success=1 default=ignore] pam_unix.so obscure sha512 rounds=640000`

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
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.65-55/8.0

Req 55	If passwords are used as an authentication attribute, "PAM" must be configured to apply password complexity rules to force the use of passwords with a minimum length of 12 characters and a combination of three out of the following categories: upper cases, lower case, numbers and special characters.

PAM must be configuration that only passwords that comply with the following complexity can be used on the system:
- Minimum length of 12 characters
- Comprising at least three of the following categories: upper/lower case letters, numbers and special characters

Motivation: Passwords with the above complexity offer high robustness against attacks coupled with acceptable user friendliness. Passwords with this level of complexity have proved their efficiency in practice. Trivial passwords that are too short are susceptible to brute force and dictionary attacks and are therefore easy for attackers to determine. Once a password has been ascertained it can be used by an attacker for unauthorized access to the system and the data on it.

Implementation example: Debian-based Distro example:
- `/etc/pam.d/common-password`contains:

```
password requisite pam_pwquality.so retry=3 minlen=12 minclass=3
enforce_for_root difok=2 maxsequence=3 maxrepeat=3
```

Note: **Suse Enterprise Linux** uses module "`pam_cracklib`", see here.

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

ID: 3.65-56/8.0

Req 56	If a password is used as an authentication attribute for technical accounts, it must have at least 30 characters and contain three of the following categories: lower-case letters, upper-case letters, digits and special characters.

Technical user accounts are characterized by the fact that they are not used by people. Instead, they are used to authenticate and authorize systems to each other or applications on a system.

A system must only use passwords for technical user accounts that meet the following complexity:
- Minimum length of 30 characters
- Comprising at least three of the following four character categories:
    - lower-case letters

- upper-case letters
- digits
- special characters

Motivation: Due to their use in machine-to-machine (M2M) communication scenarios, technical user accounts are often equipped with privileges that can be of high interest to an attacker to compromise infrastructures. Without mechanisms of extensive compromise detection, the risk of a password being determined or broken by an attacker can increase significantly over time. A significant increase in password length counteracts these risks and can also be implemented particularly easily in M2M scenarios, since handling a very long password is not a particular challenge for a machine (as opposed to a person).

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

ID: 3.01-27/8.0

Req 57	If a password is used as an authentication attribute, the reuse of previous passwords must be prevented.

A history of the previously used passwords must be recorded for each user account. When a password change is initiated for a user account, the new password must be compared with this password history. If the reuse of a password is detected, the password change must be rejected. This validation process must be implemented in the system on the basis of technical measures. If a central IAM system is used for user authentication, the implementation can be forwarded to the central IAM system or outsourced there [see also Root Security Requirements Document[i] "3.69 IAM (Identity Access Management) - Framework"].

In general, the password history should ensure that a password that has already been used can never be used again.

However, due to technical limitations, a password history cannot be recorded indefinitely in many IT/NT products. In this case, the following basic rules must be observed:

- a password that has already been used must not be reusable for a period of at least 60 days (measured from the point in time at which the affected password was replaced by another)
- in systems in which the period of at least 60 days cannot be implemented, the longest possible period must be configured. In addition, it must be confirmed by a Project Security Manager (PSM) that the configured period is still sufficient in the overall context of the system with regard to the security requirement.

Annotation:
Some IT/NT products do not offer any technical configuration parameters with which the password history can be linked directly to a time period, but only allow the definition of the number of passwords to be recorded. In such cases, the time period can alternatively be ensured by linking the following, usually generally available configuration parameters. Within the resulting policy, a user can only change his password once a day and, due to the number of passwords recorded, can reuse an old password effectively after 60 days at the earliest.

- Minimum Password Age: 1 day
- Password History: Record of the last 60 passwords used

With this implementation variant, it should be noted that the minimum age for the password should not be more than one day in order not to inappropriately restrict the user with regard to the fundamental need to be able to change the password independently at any time.

Motivation: Users prefer passwords that are easy to remember and often use them repeatedly over long periods of

time when the system allows. From the user's point of view, the behavior is understandable, but effectively leads to a considerable reduction in the protective effect of this authentication parameter. With adequate knowledge of the user or information obtained from previous system compromises, an attacker can gain access to supposedly protected user accounts. Particularly in situations in which new initial passwords are assigned centrally as part of an acute risk treatment, but users change them immediately to a previous password for the sake of simplicity, there is a high risk that an attacker will resume illegal access. It is therefore important to prevent users from reusing old passwords.

Implementation example: Debian-based Distro example:

- `/etc/login.defs`contains :

```
PASS_MIN_DAYS 1
```

- `/etc/pam.d/common-password`contains (line order is important) :

```
password required pam_pwhistory.so use_authtok enforce-for-root
retry=3 remember=60
password [success=1 default=ignore] pam_unix.so obscure sha512
use_authtok
```

ID: 3.65-58/8.0

Req 58	If passwords are used as an authentication attribute, "PAM" must be configured to have protection online attacks like brute force and dictionary attacks that hinder password guessing.

Brute force and dictionary attacks aim to use automated guessing to passwords for user and machine accounts. To prevent these kind of attacks a limitation for invalid authentication reties must be configured. It is recommended to lock an account after 5 retries for 10 minutes.

Motivation: Without any protection mechanism, an attacker can possibly determine a password by executing dictionary lists or automated creation of character combinations. With the guessed password than the misuse of the according user account is possible.

Implementation example: Debian-based Distro example:

- `/etc/pam.d/common-auth`contains:

```
auth required pam_tally2.so onerr=fail audit silent deny=5 un-
lock_time=600 even_deny_root
```

For implementation details look here: www.linux.org/docs/man8/pam_tally2.html
For newer distributions like Ubuntu 22.04 LTS module "faillock" is also a solution: man8/pam_faillock.8.html

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

ID: 3.65-59/8.0

Req 59	PAM must be configured so that Message of the day (motd) outputs do not contain any data with need of protection.

Message of the day (motd) outputs must not contain any data with need of protection like number of missing patches, used software or kernel version. The following files must be checked for such information:

- /etc/pam.d/login
- /etc/pam.d/sshd

Motivation: Data with need of protection in motd can give attackers information that are helpful for further attacks.

Implementation example: Debian-based Distro example:

```
user@server:~# sudo rm /etc/update-motd.d/*
```

For this requirement the following threats are relevant:

- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.65-60/8.0

5. host-based firewall

A host-based firewall is recommended to limit the accessibility of services, as required by Requirement 2 derived from Technical Baseline Security for IT/NT Systems.

Examples are iptables, nftables, ufw.

Req 60	If iptables, nftables or ufw is used, policies for loopback traffic must be configured.

Loopback traffic is used between server processes. A policy for traffic to the loopback network (127.0.0.0/8, ::1) must be configured for all other network interfaces.

Motivation: To prevent spoofing attacks, the loopback network must be protected from such malicious traffic.

Implementation example: As example an **Ubuntu 20.04** host with **iptables** could be configured like this:

```
$ iptables -A INPUT -i lo -j ACCEPT
$ iptables -A OUTPUT -o lo -j ACCEPT
$ iptables -A INPUT -s 127.0.0.0/8 -j DROP
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Disruption of availability

ID: 3.65-61/8.0

Req 61	If iptables, nftables or ufw is used, rules for inbound and (if used) forwarding connections must be configured.

For inbound and (if used) forwarding connections rules must be configured.

- iptables: tables`INPUT`&`FORWARD`(https://ipset.netfilter.org/iptables.man.html)
- nftables: at least a table`inet`containing a base chain hook`input`& a base chain hook`forward`(https://manpages.ubuntu.com/manpages/focal/man8/nft.8.html)
- ufw: DIRECTION`incoming`&`routed`(http://manpages.ubuntu.com/manpages/focal/man8/ufw.8.html)

Motivation: Without rules all packets will be dropped by the default policy (default drop) which will prevent network usage.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Disruption of availability

ID: 3.65-62/8.0

Req 62	If iptables, nftables or ufw is used, policies must exist for all ports in listening state.

It is necessary to configure rules for all available services (ports) on a server. If a complete reachability is not needed (for example for management services like SSH) a restriction to source IP addresses or IP networks must be implemented.

Motivation: A restriction of reachability of a network service minimizes the possible attack vector.

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

- Unauthorized use of services or resources
- Disruption of availability

ID: 3.65-63/8.0

Req 63	If iptables, nftables or ufw is used, the default policy for incoming or forwarding traffic must be to drop it.

To reject connections to unconfigured network services, the default policy for inbound and forwarded traffic must be configured to drop all packets not caught by other policies.

- iptables: tables`INPUT`&`FORWARD`(https://ipset.netfilter.org/iptables.man.html)
- nftables: type filter hook`input`&`forward`(https://manpages.ubuntu.com/manpages/focal/man8/nft.8.html)
- ufw: DIRECTION`incoming`&`routed`(http://manpages.ubuntu.com/manpages/focal/man8/ufw.8.html)

Motivation: Uncontrolled access to network services is possible without proper default policy.

Implementation example:

```
$ sudo iptables -P INPUT DROP
$ sudo iptables -P FORWARD DROP

$ sudo nft chain inet filter input { policy drop \; }
$ sudo nft chain inet filter forward { policy drop \; }

$ sudo ufw default deny incoming
$ sudo ufw default deny routed
```

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Disruption of availability

ID: 3.65-64/8.0

6. Mandatory Access Control (MAC)

Mandatory Access Control (MAC) is a method of determining which users or programs on your computer are allowed to access which files, folders, or other resources. In the case of Linux, it is a security mechanism set by an administrator. MAC can be used to create specific rules that determine what actions users or programs are allowed to perform. It allows for more granular control over your computer's security by restricting access on a "need-to-know" basis.

Recommended implementations of MAC for Linux are:

1. **SELinux (Security-Enhanced Linux)** was developed by the National Security Agency (NSA) and is included as standard in many Linux distributions, including *Red Hat Enterprise Linux (RHEL), CentOS, and Fedora*. It allows for very fine granularity in controlling access rights to files, processes, ports, and other resources.
2. **AppArmor** was developed by Canonical for *Ubuntu* and is also available in other Linux distributions. Compared to SELinux, AppArmor is often known to be easier to use and set up. It works on the basis of "profiles" that restrict the access of applications to files and system resources.

Req 64	If a system has services accessible from the Internet, it is a virtualization or container host, an implementation for Mandatory Access Control (MAC) must be used to restrict services, guest VMs or containers.

Mandatory access control (MAC) enforces additional security policies for access. A MAC solution like SELinux or AppArmor must be used on systems with a higher security demand like:

- Servers with Internet facing services.
- Host systems for hypervisor virtualization
- Host systems for container solutions

Note: On Debian based Linux (e.g. Ubuntu) and Suse Enterprise Linux AppArmor is used by default. On RedHat based Linux (e.g. RHEL, Oracle Linux, CentOS) SELinux is used. It is rec-ommended to use the pre-installed MAC solution of the used Linux OS.

Motivation: The stricter access model of MAC implementation protects services better as Linux did it by default. In case of a successful compromise of a service the attacker is limited in accessing the system.

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

ID: 3.65-65/8.0

6.1. SELinux

Req 65	If MAC implementation **SELinux** is used, it must not be disabled in bootloader configuration.

Most distributions have kernel default values to load SELinux infrastructure. Only setting kernel parameter `selinux=0` would lead to deactivation. SELinux must not be deactivated in bootloader configuration of Grub to be executed during system boot.

Motivation: The stricter access model of MAC implementations protects services better as Linux did it by default. In case of a successful compromise of a service the attacker is limited in accessing the system.

Implementation example: Fedora/RedHat based Distro example:

```
$ sudo grubby --update-kernel=ALL --remove-args 'selinux=0 en-
forcing=0'
```

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
- Disruption of availability

ID: 3.65-66/8.0

Req 66	If MAC implementation **SELinux** is used, it must be run in "enforcing" mode to actually enforce policy.

Processes with high security risk should be restricted with mandatory access control (MAC) like SELinux to reduce exploitability. Software like hypervisor, container software or network services reachable from the internet are typical examples for restriction. Using profiles SELinux restricts processes to their expected behavior. Either pre- or self-configured profiles must be used for software with high security risk.
For some Linux distributions SELinux runs per default in "permissive" mode, which leads to SELinux only logging warnings but not enforcing policy. Therefore it needs also to checked whether it is set to "enforcing" mode.

Motivation: The stricter access model of MAC implementations protects services better as Linux does by default. In case of a successful compromise of a service the attacker is limited in accessing the system.

Implementation example: See `/etc/selinux/config` for configuration. For Debian-based distros using kernel parameter (`selinux=1 security=selinux enforcing=1`) on boot is recommend too. Red Hat on the other hand only uses the config file. **In any case it is best to consult distro documentation for proper understanding and configuration!**

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
- Disruption of availability

ID: 3.65-67/8.0

Req 67	If MAC implementation **SELinux** is used, the policy must be configured.

The SELinux policy must be configured at least for "`SELINUXTYPE=targeted`" or stricter if needed. Documentation on redhat.com about "Writing a custom SELinux policy" is a good place to start.

Motivation: The stricter access model of MAC implementations protects services better as Linux did it by default. In case of a successful compromise of a service the attacker is limited in accessing the system.

Implementation example: File "`/etc/selinux/config`" contains:

```
SELINUXTYPE=targeted
```

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
- Disruption of availability

ID: 3.65-68/8.0

Req 68	If MAC implementation **SELinux** is used, SETroubleshoot and MCS Translation Service must not be installed.

The SETroubleshoot is an unnecessary daemon to have running on a server, especially if X Windows is disabled. The MCS Translation Service service is also not needed for operating SELinux. Both services must not be installed or deleted.

Motivation: Not needed software and services must not be installed on a system to reduce the amount of potentially vulnerable code.

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

ID: 3.65-69/8.0

6.2. AppArmor

Req 69	If MAC implementation **AppArmor** is used, it must not be disabled in bootloader configuration.

AppArmor must be activated in bootloader configuration of Grub to be executed during system boot.

Motivation: The stricter access model of MAC implementation protects services better as Linux did it by default. In case of a successful compromise of a service the attacker is limited in accessing the system

Implementation example: GRUB 2 configuration (`/etc/default/grub`) adds Kernel parameter for example like this:

```
GRUB_CMDLINE_LINUX="apparmor=1 security=apparmor"
```

After changing GRUB 2 configuration executing "`update-grub`" is necessary.

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
- Disruption of availability

ID: 3.65-70/8.0

Req 70	If MAC-Implementation **AppArmor** is used, its state must be enforced.

Profiles for AppArmor must be enforced for daemons with high security risk like hypervisor, container software or Internet reachable network service on the server.

Motivation: The stricter access model of MAC implementation protects services better as Linux did it by default. In case of a successful compromise of a service the attacker is limited in accessing the system.

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
- Disruption of availability

ID: 3.65-71/8.0

7. Regular Compliance Checks

This chapter includes requirements that show the default behavior of a Linux OS. These requirements must normally not be configured after installation. Nevertheless, these requirements are also important to mention as it is necessary to include them in regular compliance checks in order to identify misconfiguration and to guarantee an adequate security level during the complete lifecycle of a system.

Req 71	No legacy + entries must exist in files passwd, shadows and group.

The character "+" is used as a marker in configuration files to insert data from NIS (Network Information Service) maps. Such markers must not exist in the files '/etc/passwd', '/etc/shadows' and '/etc/group'.

Motivation: Attackers could possibly inject own code to gain privileged access if the character "+" exists in files '/etc/passwd', '/etc/shadows' and '/etc/group'.

For this requirement the following threats are relevant:
- Unauthorized modification of data

ID: 3.65-72/8.0

Req 72	A user's home directory must be owned by the user and have mode 750 or more restrictive.

A user can change the permissions of its home directory. This must be done for super-user "root" (UID 0) and user accounts (UID usually >=1000). It is not required for system accounts, which are predefined by the operating system and can be found in /etc/passwd with home directories such as "/dev", "/bin" etc. It must be validated regularly that the mode of all user home directories is 750 and the corresponding user is owner of its home directory. This is only valid for accounts of real users and excludes system accounts. An exception of this requirement are home directories for users used for SFTP in a chroot environment. In this case the home directory must be owned by root.

Motivation: If accidently a wrong user becomes owner of another user's home directory or if permissions are wrong another user can access foreign data.

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

ID: 3.65-73/8.0

Req 73	Default group for the root account must be GID 0.

The usage of GID 0 as default group for the root account is the default on most Linux systems. This must not be changed and should be checked in '/etc/passwd' file on a regular basis.

Motivation: Changing the GID for root may result in root-owned files that become accessible by non-privileged users.

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

ID: 3.65-74/8.0

Req 74	Root must be the only UID 0 account.

Any account with UID 0 has root privileges on the server. It must be validated on a regular basis that only the root account has UID 0.

Motivation: A user with restricted rights can accidently become root if UID 0 is set for its account.

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

ID: 3.65-75/8.0

Req 75	All groups in /etc/passwd must exist in /etc/group.

It must be checked on a regular basis that all groups definied `/etc/passwd` are also defined in `/etc/group` .

Sample entry from `/etc/passwd` :

```
thomas:x:1000:1000:Tho Mas,,,:/home/thomas:/bin/bash
```

The red marked number represents the users primary group and this one must exist in `/etc/group` too.

Motivation: Errors in group management can result in unspecific authentication and authorization behavior.

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

ID: 3.65-76/8.0

Req 76	No duplicate UIDs and GIDs must exist.

It must be checked on a regular basis that no duplicate UIDs and GIDs in the files '/etc/passwd' and '/etc/group' exist on the server.

Motivation: Users and groups must be assigned to unique UIDs and GIDs for accountability and to ensure appropriate access protections.

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

ID: 3.65-77/8.0

Req 77	No duplicate user or group names must exist.

The tools "useradd" and " groupadd" do not allow to set duplicate user or group names. Unfortunately, an administrator can do so by manually editing the corresponding configuration files. The file "`/etc/passwd`" and "`/etc/group`" must be checked for duplicate entries.

Motivation: Duplicate user or group name can result in unauthorized access to data.

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

ID: 3.65-78/8.0

Req 78	If a Debian-based Linux distribution is used, the "shadow" group must be empty.

On Debian-based Linux distributions the shadow group allows system programs which require access the ability to read the '/etc/shadow' file. No users must be assigned to the shadow group. The shadow group must be checked in file '/etc/group' on a regular basis that no users are assigned to it.

Motivation: With unauthorized access to the '/etc/shadow' file it is possible to run password cracking attacks against the stored password hashes.

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

ID: 3.65-79/8.0

Req 79	No files and directories without assigned user or group must exist.

If users or groups are deleted from a system, their files and directories must also be deleted, or the ownership must be transferred to another user or group. Otherwise, files and directories without a user or group are left on the system. The system must be checked for files and directories without assigned user or group on a regular basis.

Motivation: If files and directories without an assigned user or group exist on system it could happen that a newly generated user can access this data if same UID or GID is assigned as used by the user or group deleted before.

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

ID: 3.65-80/8.0

Req 80	Security relevant configuration files must have proper ownership and restrictied permission.

The permissions of configuration files must be correct and set to according user and group. This applies to files of the Shadow Password Suite (passwd,group,shadow), GRUB 2 Boatloader (grub.cfg, user.cfg), cron daemon (crontab, cron.d, etc.) and Shells.

Motivation: In configuration files data with need of protection is stored. With wrong privileges, an unauthorized user can possibly access these files and misuse data or even modify configuration.

Implementation example: Debian-based Distro example assuming `bash`:

```
sudo chown root:root /boot/grub/grub.cfg ; sudo chmod u-x,go-rwx /
boot/grub/grub.cfg
sudo chown -R root:root /etc/*cron* ; sudo chmod -R go-rwx /etc/*cron*
# if symbolic links are in cron directories
# also change ownership & permission of any files referenced
sudo chown root:root /etc/passwd* ; sudo chmod u-x,go-wx /etc/passwd*
sudo chown root:root /etc/group* ; sudo chmod u-x,go-wx /etc/group*
sudo chown root:root /etc/shadow* ; sudo chmod u-x,g-wx,o-rwx /etc/
shadow*
sudo chown root:root /etc/gshadow* ; sudo chmod u-x,g-wx,o-rwx /etc/
gshadow*
sudo chown root:root /etc/security/opasswd* ; sudo chmod u-x,go-rwx /
etc/security/opasswd*
sudo chown root:root /etc/shells ; sudo chmod u-x,go-wx /etc/shells
```

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

ID: 3.65-81/8.0

8. Technical Baseline Security for IT/NT Systems

8.1. Basic System Hardening

Req 81	Predefined user accounts that are not required must be deleted or at least disabled.

On many systems, there are predefined but unused user accounts (e.g. "guest") after the initial installation.

These predefined user accounts must be deleted or at least disabled immedately after the initial installation; if these measures are not feasible, the corresponding user accounts must be blocked for remote access. In any case, disabled or blocked user accounts must also be provided with an authentication attribute (e.g. a password or an SSH key) so that unauthorized use of such a user account is prevented in the event of a misconfiguration.

Excempt from the requirement to delete or disable predefined user accounts are user accounts that are used exclusively for internal use on the corresponding system and that are required for the functionality of one or more applications of the system. Even for such a user account, it must be ensured that remote access or local login is not possible and that a user of the system cannot misuse such a user account.

Motivation: User accounts that are predefined by default in a product are typically common knowledge and can be targeted by an attacker for brute force and dictionary attacks. If these user accounts are not needed in a specific system, their existence represents an unnecessary attack surface. A particular risk is posed by predefined user accounts that are preconfigured without a password or with a well-known standard password. Such user accounts can be misused directly by an attacker if their security hardening was missed due to the unplanned use in the specific system.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-7/8.0

Req 82	Predefined authentication attributes must be changed.

After the takeover or initial installation of a system, there are usually predefined authentication attributes (e.g. passwords, SSH keys, SSL/TLS Certificates) in the system, as assigned by manufacturers, developers, suppliers or automated installation routines.

Such predefined authentication attributes must be changed to new, individual values immediately after the takeover or installation of the system.

Motivation: Values predefined by third parties in authentication attributes cannot be trusted because they do not represent a controlled secret. Affected authentication attributes can be misused by unauthorized persons to access and compromise systems. This risk is significantly increased if commonly known default values are used for authentication attributes (e.g. a default password for the administrator user account in a particular software product).

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-8/8.0

8.2. Protecting availability and integrity

Req 83	In overload situations, the system must behave in a predictable manner.

Even comprehensive native protections may not be able to prevent a system from becoming overloaded in extreme situations.

It must therefore be ensured that, in overload situations, the system does not switch to a state that overrides security-relevant functions or properties of the system. Performance losses (e.g. the reduction of the throughput of legitimate network packets or the number of answered server requests per period) are usually unavoidable in overload situations, but the regular functional behavior of the system must be fundamentally preserved.

In extreme cases, this can mean that a controlled shutdown of the system is more acceptable than continued operation in the event of uncontrolled failure of the security functions and thus the loss of system protection.

Motivation: By means of a denial-of-service attack, an attacker can try to overload a system in a targeted manner. If such a system then reacts unpredictably or fails its regular behavior, especially with regard to its security functions, this can open up an extended attack surface for the attacker on functions and data of the system and potentially endanger other linked systems.

Implementation example: A firewall that discards its filter rules in overload situations and forwards all packets without checking would not meet the requirement. In this case, blocking all packets by shutting down the firewall would be more acceptable than failing their regular task of protecting downstream systems.

For this requirement the following threats are relevant:
- Unauthorized use of services or resources
- Disruption of availability
- Unnoticeable feasible attacks

ID: 3.01-13/8.0

8.3. Protecting data and information

Req 84	Data in need of protection must be protected against unauthorized access and modification during transmission.

The need for protection of data to be transmitted depends on its classification (e.g. according to applicable legal data privacy requirements, regulatory requirements, contractual obligations), the potential damage in the event of its misuse, and other relevant factors (e.g. transmission via public networks). The nature and extent of the protective measures must be appropriately chosen.
Authentication attributes such as passwords or tokens etc. are generally considered to be in need of protection. Data that determines the functionality and security-relevant behavior of a system (e.g. updates & patches, configuration parameters, remote maintenance, control via APIs) are also considered to be fundamentally in need of protection.

Compliance with the protection objectives of confidentiality and integrity must be consistently guaranteed during the transmission of data in need of protection.

As a rule, this requires the implementation of cryptographic methods (e.g. encryption, signatures, Hashes).
Cryptographic methods may
- be applied directly to the data before transmission, which can make subsequent transmission acceptable even via insecure channels
- be used on the transmission channel to create a secure channel and protect any kind of data passing through it
- or be implemented as a combination of both.

Cryptographic methods used for the transmission of data must be recognized and effective for this purpose and must have no known vulnerabilities.

Motivation: The transmission of data without adequate protection enables an attacker to intercept, use, disseminate, modify or remove it from transmission without authorization. This potentially opens up further attack vectors on the immediate target systems as well as connected other systems and can lead to significant failures, loss of control and damage as well as resulting penalty claims and reputational losses towards customers and business partners.

Implementation example: [Example 1]
Confidential documents are encrypted before they are sent by e-mail to the customer.

[Example 2]
An administrator configures a new cloud application over the Internet. Access is via a TLS-encrypted connection ("https").

[Example 3]
A system obtains automatic software updates from an update server. The update server delivers the software updates cryptographically signed. The system can thus validate the received software updates and reliably rule out that they have been manipulated during transmission.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Disruption of availability
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-15/8.0

8.4. Session Protection

Req 85	Sessions must be protected against unauthorized takeover ("session hijacking").

Interfaces that provide session functionality to the system must implement technical measures to prevent a legitimate user's session from being taken over and continued by an unauthorized third party.

Such protection can be achieved, for example, by implementing a combination of the following options that makes sense for the specific system:

- At the transport layer: Use of the TCP protocol (with its sequence numbers) and corresponding filter lists
- At the session layer: Use of the TLS Protocol
- At the application layer: Negotiation of a random secret session key between sender and receiver to authorize all session traffic (e.g. session ID, session cookie, session token)
- Use of cryptographic methods to protect session keys from eavesdropping or modification attacks

Motivation: Unprotected sessions can potentially be hijacked and continued by an attacker in order to exercise unauthorized access to the system in the context of the affected user.

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

ID: 3.01-16/8.0

Req 86	The system must allow users to log out of their current session.

The system must have a feature that enables the logged-in user to log out at any time. It must not be possible to resume a logged-out session without re-authenticating the user.

Motivation: A user must retain complete control over the sessions he has established in order to be able to terminate his access to a system at any time according to the situation and thus protect data and functions exposed via this access. In addition, the user must be able to assume that sessions specifically terminated by him cannot subsequently be resumed and continued by unauthorized third parties.

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

ID: 3.01-17/8.0

8.5. Authentication and authorization

Req 87	The use of system functions that require protection as well as access to internal or confidential data must not be possible without prior authentication and authorization.

The use of functions of the system that require protection as well as access to data classified as internal or confidential must only be possible after the user has been uniquely identified and successfully authenticated by means of the user name and at least one authentication attribute. In addition, it must be verified that the user is authorized to access the affected functions and data within the user role assigned to him or her in the system.

An exception to this are functions and data that may be used publicly without restriction; for example, the area of a website on the Internet where only public information is provided.

Examples of features that require prior authentication include:
- Remote access to network services (such as SSH, SFTP, web services)
- Local access to the management console
- Local use of operating system and applications

Examples of authentication features that can be used:
- Passwords
- cryptographic keys or certificates (e.g., in the form of smart cards)

This requirement also applies without restriction to any machine access to the system (here the implementation is usually carried out by using so-called M2M - "Machine-to-Machine" - user accounts).

Motivation: The unambiguous authentication and authorization of access to a system are elementary to protect functions and data from misuse.

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

ID: 3.01-19/8.0

8.6. Authentication attribute "password"

Req 88	If passwords are used as an authentication attribute, those must be stored using a suitable and approved "Password Hashing" method to protect against offline-attacks like brute force or dictionary attacks.

This requirement relates to the storage of passwords in all types of user databases, as used in this system, in order to authenticate incoming access (local or remote) by users or other systems.

If an attacker obtains the copy of a user database of the system, he is able to bring it into a fully independent environment and utilize automatized dictionary or brute force attacks to determine contained passwords. Specialized tools in combination with high computing power allow for high cracking rates in a relatively short period of time, if protective measures are insufficient. Due to the independency from the source system, such an offline attack happens unnoticed.

The following countermeasure must be implemented, since this ensures best possible protection against offline attacks:

- passwords must be stored using a cryptographic one-way function (known as "Password Hashing") that is recognized as effective for this purpose and is classified as secure at the current state of the art.

Please Note: valid password hashing algorithms are described in Security Requirement Catalog "3.50 Cryptographic Algorithms and Security Protocols".

Explicitly NOT PERMISSIBLE is:

- to store passwords in cleartext
- to store passwords in any format which can be directly backcalculated
- to store passwords using reversible encryption

Please Note:
In this context, "directly backcalculatable formats" refers to those that simply encode the password, without involving a secret key in the transformation process. Since the password will no longer show up as original cleartext after it has been processed, those formats may easily be mistaken to provide confidentiality. Effectively, they do not offer any protection. The enconding is fixed and therefore an attacker can easily make use of it to compute the original cleartext password from the encoded string.
Examples for directly backcalculatable formats are: "base64", "rot13"
"Reversible" are all encryption methods which, using the appropriate key, enable encrypted content to be transformed back into the original content. Accordingly, with reversible encryption there is always the challenge of keeping the key secure and protecting it from unauthorized access. Reversibility is a required fundamental property in many areas of encryption applications, e.g. for transferring confidential messages, but it is counterproductive for storing passwords: a stored password must remain comparable by means of technical methods, but it must no longer be possible to convert it back into plain text in order to protect it as well as possible from unauthorized viewing.
Examples for reversible encryption are: "AES", "CHACHA20", "3DES", "RSA"

Motivation: Without protective measures, an attacker in possession of a user database copy is able to determine masses of contained passwords in short time by merely trying out character string combinations or making use of dictionaries. Passwords stored in cleartext or any backcalculatable format are fully defenseless to an offline attack. Once a password has been ascertained it can be used by an attacker for unauthorized access to the system and the data on it.

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
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-24/8.0

Req 89	If a password is used as an authentication attribute, a protection against online attacks like brute force and dictionary attacks that hinder password guessing must be implemented.

Online brute force and dictionary attacks aim for a regular access interface of the system while making use of automated guessing to ascertain passwords for user accounts.

To prevent this, a countermeasure or a combination of countermeasures from the following list must be implemented:

- technical enforcement of a waiting period after a login failed, right before another login attempt will be granted. The waiting period shall increase significantly with any further successive failed login attempt (for example, by doubling the waiting time after each failed attempt)
- automatic disabling of the user account after a defined quantity of successive failed login attempts (usually 5). However, it has to be taken into account that this solution needs a process for unlocking user accounts and an attacker can abuse this to deactivate accounts and make them temporarily unusable
- Using CAPTCHA ("**C**ompletely**A**utomated**P**ublic**T**uring test to tell**C**omputers and**H**umans**A**part") to prevent automated login attempts by machines ("robots" or "bots") as much as possible. A CAPTCHA is a small task that is usually based on graphical or acoustic elements and is difficult to solve by a machine. It must be taken into account that CAPTCHA are usually not barrier-free.

In order to achieve higher security, it is often meaningful to combine two or more of the measures named here. This must be evaluated in individual cases and implemented accordingly.

Motivation: Without any protection mechanism an attacker can possibly determine a password by executing dictionary lists or automated creation of character combinations. With the guessed password than the misuse of the according user account is possible.

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

ID: 3.01-25/8.0

Req 90	If a password is used as an authentication attribute, users must be able to independently change the password anytime.

The system must offer a function that enables a user to change his password at any time.

When an external centralized system for user authentication is used, it is valid to redirect or implement this function on this system.

Motivation: The fact that a user can change his authentication attribute himself at any time enables him to change it promptly if he suspects that it could have been accessed by a third party.

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
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-29/8.0

Req 91	If passwords are used as an authentication attribute, they must not be displayed in plain text during input.

Passwords must not be displayed in legible plain text on screens or other output devices while they are entered. A display while entering must not allow any conclusions to be drawn about the characters actually used in the password.

This requirement applies to all types of password input masks and fields.
Examples of this are dialogs for password assignment, password-based login to systems or changing existing passwords.

Exceptions:

- Within an input field, an optional plain text representation of a password is permitted, provided that this plain-text representation serves a valid purpose, exists only temporarily, has to be explicitly activated by the legitimate user on a case-by-case basis and can also be deactivated again immediately by the latter.
  A valid purpose would be, for example, to allow the legitimate user an uncomplicated visual check, if necessary, that he has entered the password correctly in a login dialog before finally completing the login.
  Such an optional plain text representation of a password must remain fully in the control of the legitimate user so that he can decide on its activation/deactivation according to the situation. In the default setting of the system, the plain text representation must be deactivated.
- The typical behavior on many mobile devices (smartphones) of displaying each individual character very briefly in plain text when entering a password - in order to make it easier for the user to control input - is fundamentally permissible there. However, the full password must never be displayed in plain text on the screen.

Motivation: In the case of a plain text display, there is a risk that third parties can randomly or deliberately spy on a password via the screen output while typing.

Implementation example: When displayed on the screen, each individual character is uniformly replaced by a "*" while entering a password.

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
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-31/8.0

9. Operating Systems

9.1. System Hardening

Req 92	If services cannot be bound to the minimal required interfaces by configuration, a local (packet) filter must regulate the accessibility of the service.

Sometimes, software cannot be bound to dedicated interfaces. A local packet filter or TCP-wrapper can ensure this.

Motivation: Packet filters offer effective protection in order to prevent misuse services from other networks.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.37-3/8.0

Req 93	The privileges for processes, services and applications must be reduced to the minimum required for the tasks they have to perform.

Privileges to processes and services must be restricted to a level in which they have only access to system ressources really needed. Suitable access restrictions must also be assigned for files that are components of the operating system or of applications or that are generated by the same (e.g. configuration and logging files).

The execution of applications and parts of them must also take place with privileges that are as low as possible. Applications should not be executed with administrator or system rights. In particular, mechanisms have to be avoided that allow processes to gain increased privileges during run-time.

Motivation: If the privileges granted to a process on a system are too broad, it could be possible to access data and parts of other services/applications for which viewing or the use is not permitted. This would give the opportunity to disclose or modify confidential data and to manipulate system files. Applications with rights that are too broad can be used by a user to extend the own authorizations and thus to gain access to files and system components to which the user would not have had access with the authorizations under normal circumstances.

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

ID: 3.37-6/8.0

9.2. System Update

Req 94	Known vulnerabilities in software and hardware of a system must be fixed or protected against misuse.

Prior to installation of a software or hardware component, users must check whether any vulnerability has been discovered and published for the version they are installing. Any component that proves to have a vulnerability must not be installed or used. Excepted from this rule are components for which the vendor has already provided a measure to remedy the vulnerability, e.g. a patch, update or workaround. In this case, the additional measure must be implemented on the system.

Hint: For the vulnerability management, a continuous process during the complete life cycle of the system is needed to fix upcoming vulnerabilities promptly.

Motivation: Publication of vulnerabilities increases the risk of successful exploitation by an attacker. The likelihood raises because of publication of detailed information and tools that help to exploit the vulnerability.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Disruption of availability
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.37-16/8.0

9.3. Availability and Integrity

Req 95	Growing (dynamic) content must not influence system functions.

Growing log data and uploads must not influence system functions.

Motivation: A filled up filesystem could stop the system from operations.

Implementation example: Usage of dedicated filesystems, separated from main system functions, or quotas, or at least a file system monitoring.

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

ID: 3.37-23/8.0

9.3.1. TCP/IP Stack

Req 96	The IPv4 and IPv6 addresses of all interfaces of a server must be configured statically.

IP addresses providing services must not be changed on external influence, even in the case of an enforced reboot. An automatic assignment of IP addresses, e.g., using DHCPv4/v6 or IPv6 auto-configuration is permitted only in the case when deactivated after initial allocation or secured otherwise. IPv6 router advertisements must be ignored.

It is recommended to form the host part of the IPv6 addresses randomly. Due to the very large address space of IPv6, this way it is very time-consuming for an attacker to use scans to discover systems.

Motivation: An attacker on the same network segment can send DHCP / RA packets to redirect network traffic to its own server (Man-In-The-Middle) and so listen to all communications.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Disruption of availability

ID: 3.37-24/8.0

Req 97	The TCP/IP stack must be implemented and configured in accordance with current knowledge to prevent attacks against the system and its network connections.

Typical attacks against the TCP / IP stack must not jeopardize the stability and integrity of the system. This includes attacks such as SYN attacks, hijacking of TCP connections, Ping of Death. Corresponding kernel features and parameters have to be set appropriately for the intended system use, as certainly respected for the operating system parameters. This also includes dropping of IP packets with unnecessary options or extension headers, e.g., source routing. Normally, no IP options or extension headers are needed.

In general, the system must be robust against incorrect and unexpected data packets on the network interface. The following typical implementation mistakes must not be done:

- No validation on the lengths of transferred data

- Incorrect assumptions about data formats
- No validation that received data complies with the specification
- Insufficient handling of protocol errors in received data
- Insufficient restriction on recursion when parsing complex data formats

Motivation: Avoid potential denial of service attacks against the TCP / IP stack by exploiting vulnerabilities or forcing a resource-intensive processing of IP packets.

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

ID: 3.37-25/8.0

Req 98	Systems must not process IP packets whose source address is not reachable via the incoming interface.

The TCP / IP stack default behaviour is that IP packets can be accepted on an interface that is not the destination of the IP packet. For that it is only necessary that a route to the sender's source address exists. In certain cases (clusters, load balancers) this is an intentional effect, but not the default.

It is necessary to ensure there are no unneeded default routes which is typically the case for internal systems.

Motivation: In such a case the IP packet comes from an untrusted source (spoofed address) or a routing error exists in the network. In both cases the packet has to be dropped.

Implementation example: Use of "Reverse Path Filter" (RPF) which provides this feature.

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

ID: 3.37-26/8.0

Req 99	The processing of ICMPv4 and ICMPv6 packets which are not required for operation must be disabled.

There are different types of ICMPv4 and ICMPv6 that are not used in most networks, but represent a risk. These types must be disabled or filtered and not be answered, sent or processed.

By contrast, the following ICMP types are permitted and may be used:
- Echo Request [Type 8 (v4), Type 128 (v6)]
- Echo Reply [Type 0 (v4), Type 129 (v6)]
- Destination Unreachable [Type 3 (v4), Type 1 (v6)]
- Time Exceeded [Type 11 (v4), Type 3 (v6)]
- Parameter Problem [Type 12 (v4), Type 4 (v6)]
- Packet Too Big [Type 2 (only v6)]
- Neighbor Solicitation [Type 135 (only v6)]
- Neighbor Advertisement [Type 136 (only v6)]

It is possible that other types will be necessary. This should be checked in each individual case. The ICMPv4 types "Timestamp Reply (14)," "Netmask Reply (18)," "Information Reply (16)" and "Redirect (5)" and ICMPv6 types "Router Solicitation" (133), "Router Advertisement" (134) und "Redirect" (137) must not be responded to or processed under any circumstances.

Motivation: ICMPv4 and v6 packets can be used by an attacker to request specific information which can be helpful for planning further attacks. In addition, it may be possible to influence the availability of systems.

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

ID: 3.37-27/8.0

9.4. Authentication and Authorization

Req 100	An authentication method must be used that allows unambiguous identification of the user.

Users must be identified unambiguously by the system. This can typically be reached by using a unique user account per user.

Role-based accounts (functional account) can be used, in which individual authentication features (e.g. certificates, cryptographic keys) are neccessary.
Intended use of this is, for example, the administration account for a database or a web server, where usually several administrators need access to this account.

So-called group accounts where authentication is done by a secret known by all involved users, e.g., a password, must not be used.

So-called machine accounts will be used for identification and authorization from systems to each other or for applications on a system and can't be assigned to a single person. Such accounts must be assigned on a per system or per application basis. It has to be guaranteed that these accounts can't be misused. Possibilities to protect them are:

- Configuration of a password - fulfilling well-known security requirements - that is known by only the administrators with a need-to-know.
- Configuring the account that only a local use is possible and an interactive login isn't possible.
- Use of a technique for authentication of the specific user account with public and private key or certificates.
- Limiting the access for legitimized systems.

Additional solutions must be checked on their usability per individual case.

Motivation: Unambiguous user identification is mandatory to assign a user rights that are necessary to perform the required tasks on the system. This is the only way to adequately control access to data and services and to prevent misuse.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources
- Denial of executed activities

ID: 3.37-28/8.0

Req 101	Accounts must be protected against unauthorized use by at least one authentication attribute.

The various user, role and machine accounts on a system must be protected from misuse. To this end, an authentication attribute is typically used, which, when combined with the user name, enables unambiguous authentication and identification of the authorized user.

Authentication attributes include: Cryptographic key, Token, Password, PIN.

This means that authentication based on a parameter that can be spoofed is not permitted. Exceptions are attributes that cannot be faked or spoofed by an attacker. Two of the above options can be combined (2-factor authentication) to achieve a higher level of security.

In companies of Deutsche Telekom group where the MyCard or a comparable smartcard is available this solution should be preferred.

Motivation: Accounts that are not protected can be used by an attacker to gain unauthorized access to a system, the data and applications stored on it.

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
- Denial of executed activities

ID: 3.37-29/8.0

Req 102	There must be no privilege escalation method which allows gaining administrator/root privileges from a user account without a sufficiently strong, renewed authentication.

Privilege escalation methodes like su, sudo or runas include always the risk that more permissions are gained than needed. The number of exploits in such mechanisms shows the complexity and vulnerability of these solutions which therefore cannot be trusted.

The login into a privileged account must always be done with two authentication attributes.

Motivation: If an attacker compromises an account which has permission to a privilege escalation, it may be possible that the attacker gets access to wide parts of the system and stored data.

Implementation example: Disable insecure privilege escalation methods so that users are required to (re-)login directly into the account with the needed permissions.

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

ID: 3.37-31/8.0