Security requirement

Apache Web Servers

Deutsche Telekom Group

Version	6.0
Date	Dec 1, 2023
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
	3.36	Security requirement

Version	State	Status
6.0	Dec 1, 2023	Released

Contact	Validity	Released by
Telekom Security psa.telekom.de	Dec 1, 2023 - Nov 30, 2028	Stefan Pütz, Leiter SEC-T-TST

Summary
This document was created on the basis of the general security policies of the Group and defines the requirements for securely implementing Apache web servers. The requirements described in this document must be met to ensure that an Apache web server cannot be easily misused by competent attackers.

Table of Contents

1. Introduction

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

2. Software

Req 1	Software and hardware of the system must be covered by security vulnerability support from the supplier.

Only software and hardware products for which there is security vulnerability support by the supplier may be used in a system.

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 in place 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 limitations. Some suppliers define their support fundamentally in increments, which may include limitations even during the final phase before the absolute end date of regular support.
If a product is used within support phases that are subject to limitations, 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 them) operates a comprehensive security vulnerability management for the affected product, which meets the above-mentioned criteria and is considered to be reliably established.

Motivation: Hardware and software products for which there is no comprehensive security vulnerability support from the supplier pose a risk. This means that a product is not adequately checked to determine whether it is affected by further developed forms of attack or newly discovered vulnerabilities in technical implementations. Likewise, if there are existing security vulnerabilities in a product, no improvements (e.g. updates, patches) are provided. This results in a system whose weak points cannot be remedied, so that they remain exploitable by an attacker in order to compromise the system or to adversely affect 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
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.01-1/7.0

Req 2	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.

Suitable 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

For this requirement the following warranty objectives are relevant:

ID: 3.01-2/7.0

Req 3	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

For this requirement the following warranty objectives are relevant:

ID: 3.01-10/7.0

3. Configuration

Req 4	Web server processes must not run with system privileges.

If a process is started by a user with system privileges, execution must be transferred to a different user without system privileges after the start.
One httpd process runs as the user, who started the web server, e.g. root. This is common for Apache web servers and is accepted.

Motivation: If the web server process runs with administrative privileges, an attacker who obtains control over this process may control the entire system.

Implementation example: User and group of the webserver processes are defined in the configuration file:

```
User apache
Group apache
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-4/6.0

Req 5	The web server service must be bound only to interfaces, which are necessary to connect the service.

In most cases the web server service needs to be bound only to one interface.

Motivation: The more interfaces provide access to the web server, the higher is the attack risk.

Implementation example: Configure the appropriate IP address with the listen directive in the config file:

```
Listen 111.222.333.444:80
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-5/6.0

Req 6	Write access for web server configuration files must only be granted to the owner of the web server processs or a user with system privileges.

To enhace the security it is recommended to create an additional non privileged user, who has write access to web server configuration files and to grant only read access to the owner of the web server process. Then an attacker, who got control over the web server process, could not modify configuration files directly.

Motivation: Configuration files may only be written by the owner of the web server process or a user with system privileges. Otherwise it would be possible for unauthorized users to change the configuration of the web server or to obtain configuration information which could be used for an attack.

Implementation example: Delete “read” and “write” access rights for “others.” Only grant “write” access to the user who configures the web server.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-6/6.0

Req 7	Apache modules that are not required must be deactivated.

The following modules are loaded by default, but are not needed in most cases. Therefore they shall not be loaded, if not needed.

mod_authn*
mod_authz_dbd
mod_authz_dbm
mod_authz_groupfile
mod_authz_owner
mod_authz_user
mod_autoindex
mod_cgi*
mod_dav*
mod_info
mod_lbmethod*
mod_lua
mod_mime_magic
mod_proxy*
mod_status
mod_userdir

Motivation: Each add-on, component or function can have security vulnerabilities.

Implementation example: Modules are loaded with the directive `LoadModule` in a configuration file.
Depending on the organisation of the configuration files there are different ways for the configuration.

If the LoadModule directives are in a configuration file, they can be deactivated with a comment sign:

```
#LoadModule ...
```

If the modules are configured by links in the `mods-enabled` directory, the links can be deleted to deactivate the modules.

You can check which modules are loaded in your current configuration by following command:
RHEL based Linux: `httpd -M`
Debian based Linux: `apachectl -M`

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-7/6.0

Req 8	If Server Side Includes (SSI) are active, execution of system commands must be deactivated.

The execution of system commands for SSI can be deactivated by setting the “IncludesNoExec” option in the “Directory” directive.

Motivation: The Server Side Includes (SSI) technology, which is implemented as an additionally loadable module, can potentially be used by attackers. The “exec” function of SSI, in particular, could be used to execute system commands, which represents a risk.

Implementation example: Enter options in the configuration file in the “Directory” directive:

```
<Directory document_directory>
  Options +IncludesNoExec
  …
</Directory>
```

Replace text in italics appropriately.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-8/6.0

Req 9	HTTP methods that are not required must be deactivated.

The TRACE method must not be used by a productive web server.

Standard requests to web servers only use GET and POST.
If the web server is in front of a REST API, then PUT and DELETE may also be required.

Motivation: HTTP TRACE could be misused by an attacker. This method allows for debugging and the trace analysis for connections between the client and the web server. Other HTTP methods could also be used to obtain information about the server, or they could be directly misused by an attacker.

Implementation example: Add the following lines in the configuration file:

```
<Location />
  AllowMethods GET POST
</Location>
```

Add

```
TraceEnable off
```

to the configuration file outside the “Directory” or "Location" directives.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-9/6.0

Req 10	If symbolic links are not required, the function to follow symbolic links must be deactivated.

Motivation: By using symbolic links, it is possible to access files outside the specified document master directory. This could lead to unwanted file access. The relevant risk can be reduced by restricting access rights.

Implementation example: Enter options in the configuration file in the “Directory” directive:

```
<Directory document_directory>
  Options –FollowSymLinks
  …
</Directory>
```

Replace text in italics appropriately.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-10/6.0

Req 11	If symbolic links are required, file access must be restricted to those files, which belong to the owner of the link.

Motivation: This limitation reduces the risk, to deliver files by the web server, which are not intended to be delivered.

Implementation example: Enter options in the configuration file in the “Directory” directive:

```
<Directory document_directory>
Options +SymLinksIfOwnerMatch
…
</Directory>
```

Replace text in italics appropriately.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-11/6.0

Req 12	CGI must not be used.

Motivation: Inappropriate CGI configuration may allow multiple attack vectors. Modern web servers provide safer and more performant alternatives to CGI. Therefore CGI is neither necessary nor recommended.

Implementation example: Apache modules for CGI shall not be loaded. This is achived by removing the appropriate lines or setting them to comments.

```
# LoadModule cgid_module modules/mod_cgid.so
# LoadModule cgi_module modules/mod_cgi.so
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-12/6.0

Req 13	If WebDAV is used for writing files, access must not be granted without successful authentication.

Motivation: WebDav makes it possible to update content online which has been made available by the web server. This function could therefore be misused to change website content.

Implementation example: The authentication is configured in the example by Auth* directives:

```
DavLockDB "/usr/local/apache2/var/DavLock"

<Directory "/usr/local/apache2/htdocs/foo">
    Require all granted
    Dav On

    AuthType Basic
    AuthName DAV
    AuthUserFile "user.passwd"

    <LimitExcept GET POST OPTIONS>
        Require user admin
    </LimitExcept>
</Directory>
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-13/6.0

Req 14	If WebDAV ist used, the access to needed directories must be restricted regarding the authorized user.

Access rights to all files accessible by WebDAV must be configures as restrictively as possible. Additionally, if Web-DAV is used, WebDAV access must be restricted to the directories required.

Motivation: WebDav makes it possible to update content online which has been made available by the web server. This function could therefore be misused to change website content.

Implementation example: The following example limits access to a directory and specific users:

```
DavLockDB "/usr/local/apache2/var/DavLock"

<Directory "/usr/local/apache2/htdocs/foo">
    Require all granted
    Dav On

    AuthType Basic
    AuthName DAV
    AuthUserFile "user.passwd"

    <LimitExcept GET POST OPTIONS>
        Require user admin
    </LimitExcept>
</Directory>
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-14/6.0

Req 15	Directory listings (indexing) must be deactivated.

Motivation: Directory listings provide information on files and directory structures which could be misused.

Implementation example: Enter options in the configuration file in the “Directory” directive:

```
<Directory document_directory>
  Options –Indexes
  …
</Directory>
```

Replace text in italics appropriately.

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data

For this requirement the following warranty objectives are relevant:

ID: 3.36-15/6.0

Req 16	The web server must be robust against overload situations.

A web server must provide security measures to deal with overload situations. In particular, partial or complete impairment of web server availability must be avoided. Potential protective measures include:

- Restricting the maximum number of HTTP sessions per IP address
- Defining the maximum size of a HTTP request
- Defining a timeout for HTTP requests

Motivation: Attackers often try to bring a web server into an overload situation by using denial-of-service (DoS) attacks. If such an attack is successful, the web server's availability or integrity may be impaired.

Implementation example: Configure protecting measures with the following directives.
The stated values are examples, which may need to be adapted to a concrete situation. If, for example, files can be uploaded via the web server, the value for `LimitRequestBody` must be increased accordingly.
Detailed information on the directives can be found in the Apache documentation.

Timeout (in seconds) for receiving HTTP requests:

```
RequestReadTimeout header=10-20,MinRate=500 body=10-20,MinRate=500
```

Timeout (in seconds) for waiting on certain events of HTTP communication:

```
Timeout 30
```

Timeout (in seconds) for subsequent requests on a persistent connection:

```
KeepAliveTimeout 5
```

Limits on HTTP requests:

`LimitRequestBody 10000` (Maximum number of bytes of the body, to be increased for file upload functionality, e.g. `10000000`)
`LimitRequestFields 50` (Maximum number of request header fields)
`LimitRequestFieldSize 8190` (Maximum number of bytes of an HTTP header)
`LimitRequestLine 8190` (Maximum number of bytes of the request line)
`LimitXMLRequestBody 10000` (Maximum number of bytes of an XML-based request body)

Limit the number of connections that will be processed simultaneously:

`MaxRequestWorkers 256` (this is differently defined for threaded and non-threaded server, see the documentation)

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

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

ID: 3.36-16/6.0

Req 17	Default content must be removed.

Motivation: By using examples, information could be obtained about the installed software (version). Examples can include security vulnerabilities.

Implementation example: Default content is activated by configuration.

In Redhat/CentOS comment out all lines in the configuration file
```
/etc/httpd/conf.d/welcome.conf
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-17/6.0

Req 18	Information about the web server in HTTP headers must be minimized.

The HTTP header must not include information on the version of the web server and the modules/add-ons used.

Motivation: Any information about the web server could allow conclusions to be drawn about security vulnerabilities.

Implementation example: Set the directive ServerTokens in the configuration file:
```
ServerTokens Prod
```
Then the HTTP response header “Server” contains just “Apache”.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-18/6.0

Req 19	Web server information in error pages, which are delivered by the web server, must be removed.

Default error pages must be replaced with user-defined error pages.
User-defined error pages must not include version information about the web server and the modules/addons used.
Error messages must not include internal information such as internal server names, error codes, etc.

Motivation: Any information about the web server could allow conclusions to be drawn about security vulnerabilities.

Implementation example: Configure user-defined error pages in the configuration file with the “ErrorDocument” direct-

ive, for example:

```
ErrorDocument 400 "Bad Request"
ErrorDocument 401 Unauthorized
ErrorDocument 403 Forbidden
ErrorDocument 404 "Not Found"
ErrorDocument 405 "Method Not Allowed"
ErrorDocument 408 "Request Time Out"
ErrorDocument 410 Gone
ErrorDocument 411 "Length Required"
ErrorDocument 412 "Precondition Failed"
ErrorDocument 413 "Request Entity Too Large"
ErrorDocument 414 "Request URI Too Large"
ErrorDocument 415 "Unsupported Media Type"
ErrorDocument 500 "Internal Server Error"
ErrorDocument 501 "Not Implemented"
ErrorDocument 502 "Bad Gateway"
ErrorDocument 503 "Service Unavailable"
ErrorDocument 506 "Variant Also Varies"
```

For this requirement the following threats are relevant:

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-19/6.0

Req 20	Unauthorized overwriting of the web server configuration must be prevented.

Motivation: By injecting an .htaccess file through an unauthorized upload, it would be possible for an attacker to override configuration directives of the web server. This can be effectively prevented by configuring AllowOverride.

Implementation example: In the configuration file in the “Directory” directive:

```
<Directory document_directory>
  AllowOverride None
  ...
</Directory>
```

Replace text in italics appropriately.

For this requirement the following threats are relevant:

- Unauthorized modification of data

For this requirement the following warranty objectives are relevant:

ID: 3.36-20/6.0

4. HTTPS

Req 21	Data in need of protection must be protected against unauthorized access and modification during transmission.

In the case of the Apache web server, the transmission of data in need of protection is about the user data of the application.

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: The data is transmitted via a TLS-encrypted connection ("https").

ID: 3.36-21/6.0

Req 22	For encryption with HTTPS the TLS protocol in version 1.2 or higher must be used.

SSL and TLS 1.0/1.1 must be considered outdated and thus may not be activated or must be deactivated, respectively. TLS in version 1.2 provides a sufficient protocol security and also offers Authenticated Encryption Associated Data (AEAD) encryption schemes.

Motivation: The current versions of TLS fix previous known security vulnerabilites and attack surfaces on the TLS protocol handshake.

Implementation example: Define TLS protocols v1.2 and v1.3 in the configuration file:

```
SSLProtocol -all +TLSv1.2 +TLSv1.3
```

TLSv1.3 ist supported by openssl v1.1.1 or higher.
With the command "`openssl version`" the used openssl version is shown.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-22/6.0

Req 23	The web server must be configured in such a way that the use of the latest version of the TLS protocol is enabled.

The latest version of the protocol offers the best possible protection and contains fixes to known vulnerabilities in previous versions of the protocol.

Motivation: The latest version of the protocol offers the best possible protection and contains fixes to known vulnerabilities in previous versions of the protocol.

Implementation example: It depends on the module used for TLS, which version is supported.
mod_ssl uses OpenSSL, which supports TLSv1.3 since version 1.1.1.

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-23/6.0

Req 24	The TLS configuration must use secure cipher suites.

Acceptable cipher suites may only use the following algorithms:

Server/Client Authentication & Key Agreement	Encryption	Message Authentication & Integrity (MAC)
ECDHE_ECDSA	AES_128_CBC[2]	SHA256
ECDHE_RSA	AES_128_GCM	SHA384
DHE_DSS[1]	AES_128_CCM	SHA512
DHE_RSA[1]	AES_192_CBC[2]	SHA-3-256
	AES_192_GCM	SHA-3-384
	AES_192_CCM	SHA-3-512
	AES_256_CBC[2]	
	AES_256_GCM	
	AES_256_CCM	
	CHACHA20_POLY1305	

[1] min. 4096-bit Parameter
[2] CBC is accepted for existing systems only

TLS 1.3 explicitly specifies the usage of only DHE and ECDHE for server/client authentication and key agreement. Thus TLS 1.3 cipher suite notation does not contain an indication in this regard.

By fulfilling this requirement the Perfect Forward Secrecy (PFS) property in the TLS/SSL implementation will be achieved.

The TLS configuration required here may no longer be secure due to new findings on cipher suites or the TLS protocol. It is recommended to check your own configuration using the following information sources and tools:

- Cipher Suite Info (https://ciphersuite.info/)
- SSL Labs (https://www.ssllabs.com/ssltest/)
- Testssl (https://testssl.sh/)
- DTSP Internal Scan Platform (https://dtsp.telekom-dienste.de/)

Motivation: Cipher suites known to be unsecure do not offer sufficient protection.

Implementation example: Define allowed ciphers in the configuration file:

```
SSLCipherSuite 'EECDH:EDH:!SHA'
```

Alternative: ECDHE with AES-GCM ciphers only:

```
SSLCipherSuite 'EECDH+AESGCM:!SHA'
```

Tip:
To determine which ciphers are enabled with that configuration, you can use `openssl` with the `-v` parameter. Example:

```
openssl ciphers -v 'EECDH+AESGCM:!SHA'
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-24/6.0

Req 25	SSL compression must be deactivated.

Motivation: The SSL compression contains a vulnerability, which allows the "crime" attack.

Implementation example: A configuration is not necessary for apache versions higher than 2.4.3, because SSL compression is deactivated by default. The configuration file must not contain

```
SSLCompression on
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-25/6.0

Req 26	The TLS configuration must provide that the cipher suite considered most secure is being chosen with highest priority.

A cipher suite contains the definition of four algoritthms. These are used for key exchange, authentication, encryption and as a hash function. General guidelines for the prioritization are

- For the key exchange the Diffie-Hellman method must be preferred because it offers perfect forward secrecy. Cipher suites using the Diffie-Hellman method usually may be identified by the strings DHE or ECDHE. ECDHE has higher priority than DHE.
- For encryption the Advanced Encryption Standard (AES) or Camellia with a key length as big as possible has to be used.
- As a hash function SHA-2 has to be used. This function usually may be identified by the string SHA followed by a number (256, 384 or 512). Warning: if the string SHA is not followed by a number this identifies the SHA-1 function which is significantly less secure.

Motivation: When a TLS connection is being established a cipher suite is selected based on the cipher suites available both on client and on server side. In order to ensure a high compatibility to all kinds of client systems the web server must not only allow for the cipher suites considered most secure. To make sure that nevertheless for each client the best possible cipher suite is selected and thus the connection is best protected the configuration must contain an according prioritization.

Implementation example: Activate the option SSLHonorCipherOrder in the confguration file:

```
SSLHonorCipherOrder On
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.36-26/6.0

Req 27	Certificates must be issued by a certification authority whose certificates are recognized by the commonly used web browsers.

For critical applications that can be used via the Internet, use of an extended validation certificate (EV certificate) is recommended.

Motivation: Only if the certificate authority (CA) is contained in the CA list of the browser being used the browser can verify the authenticity of the server.or web application
Stricter issuing criteria apply to EV certificates. If an EV certificate is used, this is visualized in the browser. Even if EV certificates do not improve security, their use increases the trustworthiness of the server for the user.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-25/6.0

Req 28	Certificates must lose their validity after a maximum of 1 year.

In the case of certificates of an internal CA, in particular for machine interfaces, the period may be extended to a maximum of 3 years.

Motivation: The methods used for analysing and breaking cryptographic processes are improved continuously. Therefore the security of the certificates can be ensured for a limited period only. But, according to a general estimation, the security of the certificates is ensured for the required validity period of one year, if an appropriate key length is used.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-26/6.0

Req 29	Certificates must have a key length of at least 3072 bits when using RSA or 256 bits when using ECC.

Remarks on DSA and RSA certificates:
For DSA and RSA, key lengths smaller than 3000 bits may only be used in legacy systems [BSI TR-02102-1] until the end of 2025 and
should be substituted at the next opportunity. Because of the better performance, elliptic curve (EC-DSA) certificates shall be preferred (if supported and technically doable).
RSA-PKCS#1 v1.5 may only be used in legacy systems and should be (if feasible) substituted at the earliest opportunity [BSI TR-02102-1].

References:
[BSI TR-02102-1] Bundesamt für Sicherheit in der Informationstechnik: Cryptographic Mechanisms: Recommendations and Key Lengths, TR-02102-1, Version 2022-01, 28.01.2022

Motivation: In order to guarantee the security of certificates over the validity period, the cryptographic keys must have an appropriate length. According to a general estimation, a key length of 3072 bits provides sufficient protection for the next years. For ECC algorithms, shorter key lengths already provide the same level of security.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-27/6.0

5. Logging

Req 30	Access to the webserver must be logged.

The web server log must contain the following information:

- Access timestamp
- Source (IP address)
- Account (if known)
- URL
- Status code of web server response

Logging must be done considering the currently valid legal, wage and company regulations. This regulations state among others that logging of events can be done only earmarked. Logging of events for doing a work control of employees is not allowed.

Motivation: For the analysis of security incidents it is very important to have basic information on how the attack has been carried out. Since a webserver represents an external interface certain information about an attack is only available on the webserver, even if the attack is aimed at a downstream system. Thus logging on a web server is mandatory.

Implementation example: Logging of accesses is activated in the default configuration.
A typical configuration for the access log might look as follows:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
```

For this requirement the following threats are relevant:

- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.36-30/6.0

Req 31	Security-relevant logging data must be forwarded to a separate log server immediately after it has been generated.

Logging data must be forwarded to a separate log server immediately after it has been generated. Standardized protocols such as Syslog, SNMPv3 should be preferred.

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.

For this requirement the following threats are relevant:

- Unauthorized modification of data
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.01-35/7.0

Req 32	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: Taking into account the current legal situation and applicable data privacy regulations, the following deletion periods for locally stored security-relevant logging data are implemented on an exemplary telecommunications system:

- 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

For this requirement the following warranty objectives are relevant:

ID: 3.01-34/7.0

Req 33	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

For this requirement the following warranty objectives are relevant:

ID: 3.01-36/7.0

Req 34	The system must provide logging data that is required to detect the system-specific relevant forms of attack in a SIEM.

The forms of attack that are typically to be expected for the present system must be systematically analyzed and identified.
The MITRE Attack Matrix (https://attack.mitre.org) can be used as a structured guide during such an identification.

It must be ensured that the system generates appropriate logging data on events that are or may be related to these identified forms of attack and that can be used to detect an attack that is taking place.

The logging data must be sent to a SIEM immediately after the system event occurs.
SIEM (Security Information & Event Management) solutions collect event log data from various source systems, correlate it and evaluate it automatically in real time in order to detect anomalous activities such as ongoing attacks on IT/NT systems and to be able to initiate alarms or countermeasures.
The immediate receipt of system events is therefore absolutely crucial for the SIEM to fulfill its protective functions.

Note:
The immediate need to connect a system to a SIEM is specifically regulated by the separate "Operation" security requirements catalogs.
If the present system does not fall under this need, the requirement may be answered as "not applicable".

Motivation: A SIEM as an automated detection system for attacks can only be effective if it continuously receives sufficient and, above all, system-specific relevant event messages from the infrastructures and systems to be monitored. General standard event messages may not be sufficient to achieve an adequate level of detection and only allow rudimentary attack detections.

Implementation example: An example system allows end users to log in using a username and password. One of the typical forms of attack for this system would be to try to discover and take over user accounts with weak or frequently used passwords by means of automated password testing (dictionary or brute force attack). The example system is configured to record every failed login event in system protocols ("logs"). By routing this logging data in parallel to a SIEM, the SIEM can detect in real time that an attack is obviously taking place, alert it and thus enable immediate countermeasures.

ID: 3.01-37/7.0