**STANDAR TEKNIS KEAMANAN APLIKASI WEB**

**DI LINGKUNGAN**

**{{NAMA ORGANISASI}}**

{{LOGO ORGANISASI}}

A blue text on a white background

AI-generated content may be incorrect.

|  |  |  |
| --- | --- | --- |
| No Dokumen | : |  |
| Revisi | : |  |
| Tanggal Terbit | : |  |

|  |  |  |
| --- | --- | --- |
| Disusun  oleh : | Diketetahui  oleh: | Disetujui / Disahkan  oleh : |
|  |  |  |
| **Staf / Nama**  Jabatan | **Nama**  Jabatan | **Nama**  Jabatan |

**About TSS-WEB**

This document consists of best practices that can be used in a security standard for web-based applications and services. It may be used as a template for a custom organization-specific security standard or just a collection of suggestions of baseline requirements for teams and projects. All requirements in this document are based on common best practices in combination with our own experiences in this field.

The latest version of this document is available at https://tss-web.secodis.comin various file formats and languages. You’ll also find a way to contact the authors of this document on this site that you can use if you have any questions, suggestions or feedback.

**License**

This document is licensed under Creative Commons By 4.0 International License [(http://creativecommons.org/licenses/by/4.0)](http://creativecommons.org/licenses/by/4.0). Any duplication, distribution and changes to it for internal proposes are permitted. Changed versions of this document don’t have to be distributed under the same license (no copy left or share a like) but must reference the author and source of this document.

**Matthias Rohr**Secodis GmbH

**Document History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Changes** |
| 1.0 | 20/02/2015 | Initial release (German) |
| … | … | … |
| 1.3 (Draft) | 29/08/2016 | Initial English version based on a complete review of the standards with a lot of improvements and corrections. |
| 1.3.1 (Draft) | 28/11/2016 | * Changes to CSP statements * CVSS scoring added to the security of 3rd party components |
| 1.4 (Draft) | 31/03/2016 | * New requirements for handling X.509 certificates (8.17) * Changes for agile Development (chapter 5) * “Use of Secure JavaScript APIs” moved from 8.4 (“Output Validation”) to 8.14 (“Client-Side Security”) * Various changes to chapter 6 ("Security Tests) * Security Auditor renamed in Security Officer |
| 1.5 | 15/08/2017 | * Security officer renamed to IT security function * OWASP Top Ten was removed and will be maintained on the web site (tss-web.secodis.com) in the future. * Protection class has been renamed into assurance class * Multiple modifications of chapter 5 (“security in development process”) * Multiple modifications of chapter 7 (“supplier requirements”) * Minor modifications to file upload requirements * Requirements for HPKP removed and for referrer policy added |
| 1.6 | 19/03/2019 | * Section 1.3 (“Roles”): Changes of role description “Security Champion” and role “developer” added * Chapter 3 (“Operational Requirements”): Extension of DMZ restrictions and security monitoring * Multiple modifications of chapter 5 (“Protection of Source and Program Code”) * 8.2 (“Input Validation”): New requirement for secure object sterilization * 8.5 (“Authentication & Registration of Users): Deprecated NIST standard replaced with current NIST SP 800-63B * 8.8 (“Authentication at Backend”): Multiple modifications * 8.11 (“Access Controls”) Modification of OAuth requirements. Move of CORS and OAuth requirements to Service Security * 8.13 (“Management of Technical Keys & Secrets”) Multiple Modifications * 8.16 (“XML Parser Security”) Multiple modifications, renaming to “Service Security”; XML parser security requirements moved to 8.2 (“Input Validation”). * Appendix A: Integration von SameSite cookies and modification of CSP requirements |
| 1.7 | 24/05/2019 | * Role “Security Champion” extended * 3rd party component renamed to 3rd party dependencies and relevant requirements in chapter 4 completely revised. * Chapter 5 (“Security of Development Process”) revised * Chapter 6 (“Security Tests”) revised, and test policy aligned to assurance classes * Update of security logging and monitoring requirements |
| 1.8 | 01/07/2019 | * Chapter 3 (“Operational Requirements”) revised and made more compliant to container and cloud environments. * Chapter 5 (“Security of Development Process”) completely revised and restructured * Chapter 6 (“Security Tests”) completely revised and and restructured * Name of the standard changed to “web development” standard instead of “web application” standard. * Minor updated in other chapters. |
| 2.0 | 01/11/2020 | Complete revision, including chapters:   * Chapter 3 („Secure Operation“) * Chapter 5 („Secure Development Process“) * Chapter 6 („Security Tests“) * Section 8.11. („Data Security & Cryptography“) * Section 8.14 („Service & API Security”) |

STANDAR TEKNIS KEAMANAN APLIKASI WEB

**{{NAMA ORGANISASI DISINI}}**

Version: 1.0.0

Klasifikasi: INTERNAL

**Daftar Isi**

[1 PENDAHULUAN 3](#_Toc123739103)

[1.1 Ruang Lingkup 3](#_Toc123739104)

[1.2 Tipe Kebutuhan 3](#_Toc123739105)

[1.3 Definisi 3](#_Toc123739106)

[2 Keamanan Operasional 5](#_Toc123739107)

[3 Implementasi Panduan 6](#_Toc123739108)

[3.1 Autentikasi 6](#_Toc123739109)

[3.2 Manajemen Sesi 7](#_Toc123739110)

[3.3 Kontrol Akses 7](#_Toc123739111)

[3.4 Validasi Input 7](#_Toc123739112)

[3.5 Fungsi Kriptografi pada Verifikasi Statis 8](#_Toc123739113)

[3.6 Penanganan Eror dan Pencatatan Log 8](#_Toc123739114)

[3.7 Proteksi Data 9](#_Toc123739115)

[3.8 Keamanan Komunikasi 9](#_Toc123739116)

[3.9 Pengendalian Kode Berbahaya 9](#_Toc123739117)

[3.10 Logika Bisnis 10](#_Toc123739118)

[3.11 *File* 10](#_Toc123739119)

[3.12 Keamanan API dan *Web Service* 10](#_Toc123739120)

[3.13 Keamanan Konfigurasi 12](#_Toc123739121)

[Appendix A: Requirements for HTTP Security Header 13](#_Toc123739122)

**Document Properties**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Location |  | **tbd** |  | | |
|  |
| Owner |  | **tbd** |  | | |
|  |
| Type |  | **Technical Standard** | |  | |
|  | |
| Classification |  | **For internal use only (INTERNAL)** | | |  |
|  | | |
| Next Review |  | **tbd** |  | | |
|  |

**Referenced Documents**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document** | **Location** | | |
| TSS-WEB version 2.0 (Template) | https://tss-web.secodis.com | | |
| Information Security Policy |  | tbd |  |
|  |
| Password Policy |  | tbd |  |
|  |

**Contacts for this Document**

|  |  |  |
| --- | --- | --- |
| **Name** | **E-Mail** | **Section** |
|  |  |  |
|  |  |  |

**Document History**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Version** | | | **Date** | |  | **Changes** | | |  | **Changed By** | | |
|  | 0.1 |  |  | 20/12/2020 |  |  | Initial document based on TSS-WEB v2.0 | |  |  | John Doe |  |
|  |  |  |  |  |
|  |
|  | | |  | |  |  | | |  |  | | |

# PENDAHULUAN

This standard describes a baseline of security requirements for web-based applications, application components and services of {{nama instansi}} These requirements are mandatory for newly developed applications and recommendations for existing ones. The baseline defined in this document may be increased if necessary for applications with higher protection requirements.

## Ruang Lingkup

This standard applies for all **web development projects** that has been started after **[DD].[MM].[YYYY].** For all others of this kind, the requirements of this document are recommendations if not explicitly specified differently.

## Tipe Kebutuhan

Terdapat dua tipe kebutuhan pemenuhan dalam dokumen standar ini, yaitu:

* **Mandatori**: Menggunakan istilah “HARUS“ dan “TIDAK DIPERBOLEHKAN”.
* **Rekomendasi**: Menggunakan istilah “DIREKOMENDASIKAN”

## Definisi

This standard uses the following definitions:

* **3rd Party Dependency**: Here: 3rd party software artifacts, used by an *application* (e.g. libraries, Maven artifacts).
* **Application:** Here: Synonym for -> web application.
* **Change:** Every change to an application in production.
* **Confidential Data**: Data, which consists of
  1. confidential information (e.g. trademarks, sensible business logic, passwords or personal data),
  2. is explicitly declared as those or
  3. is only accessible by a restricted number of people.
* **Confidential Source or Program Code:** Source or program code which may consist -> *confidential data*.
* **Criticality**: Here: Mostly synonym for business criticality.
* **Dependency Repository**: System that manages *3rd party dependencies* (e.g. libraries, Maven artifacts). A dependency repository is often part of a general software repository system such as Nexus or Artifactory.
* **External Web Application**: A web-based application that is accessible from the outside of the organization (e.g. via the Internet).
* **Internal Source or Program Code**: Source or program code which is not confidential and not public (standard).
* **Internal Web Application:** A web-based application that is only accessible from the inside of the organization (e.g. intranet application).
* **Service:** Here: Synonym for web-based service or (REST) API.
* **Source Code Repository**: System where custom code is stored (e.g. SVN, Git).
* **Web Application:** Here: A software program (UI, service or API or combination of them) that is accessible via HTTP(s) protocol and fulfills a particular business case.
* **Web-based Application:** See web application.

# Keamanan Operasional

The following requirements apply to systems (infrastructure, platforms or other runtime environments) on which productive applications of {{nama instansi}} are executed:

1. **Pemisahan Lingkungan Operasional Aplikasi**: Aplikasi web *production* HARUS terpisah dari lingkungan *development* dan *testing*.
   1. TIDAK DIPERBOLEHKAN adanya komunikasi antar aplikasi web yang berbeda lingkungan operasional,
   2. Data pada lingkungan *production* TIDAK DIPERBOLEHKAN digunakan pada lingkingan *development* dan *testing.*
2. **Penguatan Sistem**: Sistem seperti server web, server aplikasi, *container platform* atau CMS, *cloud platform* atau lingkungan *production* lainnya HARUS terdapat penguatan keamanan sesuai dengan *best practices,* seperti:
   1. Penguatan Sistem Operasi,
   2. Hanya menjalankan *services*, *plugins*, dan fungsi lainnya yang dibutuhkan,
   3. Menerapkan *security headers* yang merujuk pada Lampiran A: Persyaratan HTTP *Security Header*.
   4. Menghilangkan konten dan halaman bawaan (*default*)pada sistem.
   5. Hanya menjalankan HTTP *methods* yang diperlukan (mis. GET dan POST) pada konfigurasi server.
   6. Server web dan server aplikasi TIDAK DIPERBOLEHKAN menampilkan versi dari server tersebut (mis. HTTP *response header* “Server” atau “X-Powered-By”).

# Implementasi Panduan

Panduan teknis penerapan kontrol keamanan aplikasi web;

## Autentikasi

1. Menggunakan manajemen kata sandi untuk proses autentikasi;
   1. Kata sandi pengguna HARUS sesuai dengan kebijakan kata sandi pada saat pendaftaran serta fungsi ubah kata sandi dan lupa kata sandi.
   2. Kata sandi pengguna HARUS meliputi unsur sebagai berikut:
2. Panjang kata sandi minimal memuat 8 karakter,
3. Terdiri dari huruf besar, huruf kecil, angka dan karakter khusus,
4. Tidak boleh identik dengan *username,*
5. Menggunakan *tag* HTML dengan atribut *password*,
6. Tidak boleh tercatat atau tersimpan di-*cache,*
7. Dienkripsi ketika dikirimkan melalui saluran yang tidak aman,
8. Tidak ditransmisikan menggunakan URL, dan
9. Disimpan menggunakan *salted secure* *hash* dengan algoritmaSHA-256 atau bcrypt.
   1. Kata sandi *default* pengguna HARUS diubah ketika login pertama.
   2. Kata sandi standar (ditetapkan oleh vendor) HARUS diganti dengan kata sandi sesuai poin 1.2 a, b, dan c.
   3. Ubah kata sandi:

a. Pengguna HARUS dapat mengubah kata sandi,

b. Pengguna DIREKOMENDASIKAN dapat melihat kekuatan kata sandi yang digunakan ketika mengubah kata sandi,

c. Pengguna HARUS memasukkan kata sandi lama,

d. Pengguna HARUS mengkonfirmasikan kata sandi baru yang digunakan,

e. Pengguna DIREKOMENDASIKAN diberitahu ketika kata sandi mereka telah berubah (misalnya melalui *e-mail* atau nomor telepon*)*.

* 1. Lupa kata sandi:

HARUS diotorisasi oleh pengguna dengan faktor kedua yang sama yang digunakan sebagai faktor kedua untuk identifikasi pengguna (misalnya alamat *e-mail* ataunomor telepon).

HARUS menggunakan OTT (*One Time Token*) dengan validitas terbatas yang dikirim sebagai faktor kedua (misalnya alamat *e-mail* atau nomor telepon yang didaftarkan pengguna),

HARUS menerapkan OTT yang dapat di validasi danberlaku paling lama 60 menit setelah permintaan lupa kata sandi.

* 1. HARUS menerapkan verifikasi kata sandi pada sisi server;
  2. HARUS mengatur masa berlaku dari kata sandi maksimum 6 bulan.

1. Jumlah maksimum kesalahan dalam pemasukan kata sandi HARUS tidak boleh lebih dari 6 kali;
2. Pada saat terjadi kesalahan pemasukkan autentikasi, HARUS menampilkan informasi umum, seperti “Username atau kata sandi tidak sesuai”;
3. HARUS menggunakan jalur komunikasi yang diamankan untuk proses autentikasi.

## Manajemen Sesi

1. HARUS menggunakan pengendali sesi yang disediakan oleh *framework* (kerangka kerja) aplikasi;
2. Pembuatan token sesi HARUS bersifat acak;
3. Validasi dan pencantuman session ID HARUS:
4. 1. Dikirimkan secara terenkripsi (melalui TLS/HTTPS) dalam jaringan yang tidak aman,
   2. Diperbarui setiap autentikasi pengguna berhasil,
   3. TIDAK dikirimkan melalui URL.
5. *Session cookies* HARUS menerapkan atribut keamanan httpOnly”, secure”, dan SameSite”,
6. Sesi server yang diautentikasi:
   1. HARUS dihancurkan setelah pengguna berhasil login,
   2. HARUS dihancurkan setelah pengguna yang diautentikasitidak aktif selama lebih dari 30 menit (idle atau *soft logout*),
   3. HARUS dihancurkan setelah sesi pengguna telah aktif lebih dari 24 jam,
   4. HARUS hanya ada sekali per pengguna. Saat pengguna masuk, semua objek sesi yang ada dari pengguna tersebut dihancurkan.
7. Semua operasi perubahan status (*create, update, delete*) pada sesi pengguna yang diautentikasi HARUS dilindungi dari *session replay* dan *Cross-Site Request Forgery* (CSRF).
   1. Menggunakan mekanisme perlindungan anti CSRF,
   2. HARUS menggunakan perlindungan CSRF yang disediakan oleh *framework* (kerangka kerja) aplikasi,
   3. Operasi perubahan status TIDAK DIPERBOLEHKAN melalui HTTP GET.
8. DIREKOMENDASIKAN terdapat mekanisme persetujuan pengguna.

## Kontrol Akses

1. HARUS menyediakan otorisasi pengguna untuk membatasi kontrol akses;
2. Setiap proses dan peran HARUS diterapkan seketat mungkin sesuai dengan kebutuhan bisnis tertentu;
3. HARUS menyediakan antarmuka pada sisi administrator otorisasi pengguna;
4. HARUS melakukan verifikasi kebenaran token Ketika mengakses data dan informasi yang dikecualikan.

## Validasi Input

1. HARUS menerapkan fungsi validasi input pada sisi server;
2. Validasi sisi klien DIREKOMENDASIKAN diterapkan tetapi hanya untuk alasan kegunaan selain validasi sisi server atau untuk mencegah serangan sisi klien;
3. Semua parameter input yang tidak tepercaya (misalnya antarmuka eksternal) HARUS divalidasi secara terbatas;
4. Validasi input HARUS sesuai dan memperhatikan kebutuhan tipe, panjang, dan ukuran data;
5. HARUS menerapkan mekanisme penolakan input jika terjadi kesalahan validasi;
6. HARUS memastikan server web aplikasi tidak rentan terhadap serangan validasi input;
7. DIREKOMENDASIKAN melakukan validasi positif pada seluruh input (*whitelisting*);
8. Validasi parameter aplikasi yang tidak dapat diedit (tidak ada input pengguna) DIREKOMENDASIKAN dilakukan secara implisit melalui pemeriksaan sisi server;
9. HARUS melakukan pelindungan terhadap akses yang mengandung konten skrip seperti menerapkan *integrity checking*
10. HARUS menerapkan pelindungan dari serangan injeksi basis data (*SQL Injection*).

## Fungsi Kriptografi pada Verifikasi Statis

1. HARUS menggunakan algoritma kriptografi, modul kriptografi, protokol kriptografi, dan kunci kriptografi sesuai dengan rekomendasi standar keamanan nasional/internasional yang berlaku;
2. HARUS melakukan autentikasi data yang dienkripsi;
3. Menerapkan manajemen kunci kriptografi;
4. Membuat angka acak yang menggunakan generator angka acak kriptografi;
5. Server HTTPS HARUS hanya mendukung sandi dan protokol aman. Yang tidak aman (misalnya cipher SSLv2 dan RC4) HARUS dinonaktifkan;
6. Data rahasia HARUS hanya dikirim dengan HTTP *request body* (misalnya melalui HTTP POST) tetapi tidak dalam UR;
7. Transmisi HARUS dengan HTTPS menggunakan sertifikat yang valid;
8. HARUS menggunakan header HTTP Strict Transport Security (HSTS);
9. Data rahasia HARUS dienkripsi sebelum disimpan di sisi klien atau di lingkungan *cloud* eksternal;
10. Koneksi HTTPS eksternal HARUS menggunakan sertifikat X.509 valid yang dikeluarkan oleh *Certificate Authority* (CA); dan
11. Sertifikat X.509 HARUS menggunakan RSA dengan >= 2048 bit atau ECC dengan >= 256 bit.

## Penanganan Eror dan Pencatatan Log

1. HARUS mengatur konten pesan umum dan tidak spesifik yang ditampilkan ketika terjadi kesalahan;
2. HARUS menggunakan metode penanganan eror untuk mencegah kesalahan terprediksi dan tidak terduga serta menangani seluruh pengecualian yang tidak ditangani;
3. HARUS tidak mencantumkan informasi yang dikecualikan dalam pencatatan log;
4. HARUS mengatur cakupan log yang dicatat untuk mendukung upaya penyelidikan ketika terjadi insiden;
5. HARUS mengatur pelindungan log aplikasi dari akses dan modifikasi yang tidak sah;
6. DIREKOMENDASIKAN melakukan enkripsi pada data yang disimpan untuk mencegah injeksi log;
7. HARUS melakukan sinkronisasi sumber waktu sesuai dengan zona waktu dan waktu yang benar;
8. Informasi yang DIREKOMENDASIKAN dicatat log sebagai dari bagian peristiwa keamanan:
   1. Tag keamanan (“SEC”),
   2. *Timestamp,*
   3. Subjek (ID pengguna, IP sumber),
   4. Deskripsi (akses sensitif/perubahan ke objek),
   5. Komponen yang relevan,
   6. Hasil.
9. Log teknis HARUS tidak berisi *Personal Identifiable Information* (PII).

## Proteksi Data

1. HARUS melakukan identifikasi dan penyimpanan salinan informasi yang dikecualikan (*backup* berkala);
2. HARUS melakukan pelindungan dari akses yang tidak sah terhadap informasi yang dikecualikan yang disimpan sementara dalam aplikasi;
3. Melakukan pertukaran, penghapusan, dan audit informasi yang dikecualikan;
4. Melakukan penentuan jumlah parameter;
5. Memastikan data disimpan dengan aman;
6. Menentukan metode untuk menghapus dan mengekspor data sesuai permintaan pengguna; dan
7. Membersihkan memori setelah tidak diperlukan.

## Keamanan Komunikasi

1. Menggunakan komunikasi terenkripsi;
2. Mengatur koneksi masuk dan keluar yang aman dan terenkripsi dari sisi pengguna;
3. Mengatur jenis algoritma yang digunakan dan alat pengujiannya; dan
4. Mengatur aktivasi dan konfigurasi sertifikat elektronik yang diterbitkan oleh penyelenggara sertifikasi elektronik.

## Pengendalian Kode Berbahaya

1. Menggunakan analisis kode dalam kontrol kode berbahaya;
2. HARUS memastikan kode sumber aplikasi dan pustaka tidak mengandung kode berbahaya dan fungsionalitas lain yang tidak diinginkan;
3. Mengatur izin terkait fitur atau sensor terkait privasi;
4. Mengatur pelindungan integritas; dan
5. Mengatur mekanisme fitur pembaruan.

## Logika Bisnis

1. Memproses alur logika bisnis dalam urutan langkah dan waktu yang realistis
2. Memastikan logika bisnis memiliki batasan dan validasi;
3. Memonitor aktivitas yang tidak biasa;
4. Membantu dalam kontrol antiotomatisasi; dan
5. Memberikan peringatan ketika terjadi serangan otomatis atau aktivitas yang tidak biasa.

## *File*

1. HARUS mengatur jumlah file untuk setiap pengguna dan kuota ukuran file yang diunggah;
   1. Maksimum ukuran file sebesar 5 MB),
   2. Memiliki pembatasan jumlah file yang diunggah sesuai dengan kebutuhan sistem.
2. File yang diunggah DIREKOMENDASIKAN tersimpan di *database*;
3. File yang diunggah HARUS dipastikan tersimpan dengan akses terbatas (mis. Pada server Linux „chmod 0600“);
4. File yang diunggah DIREKOMENDASIKAN tersimpan pada file baru dengan nama file yang unik dan tidak menunjukkan informasi pribadi pengguna;
5. HARUS melakukan validasi file sesuai dengan tipe konten yang diharapkan;
   1. File yang diunggah tidak mengandung kode yang dapat tereksekusi (mis. ,”.html”, “.php”, “.js”);
   2. Menentukan tipe file yang dapat diupload,
   3. Melakukan validasi ekstensi file dan MIME *type.*
6. DIREKOMENDASIKAN melakukan konfigurasi server dengan *whitelist* ekstensi file yang boleh diunggah.

## Keamanan API dan *Web Service*

1. Layanan eksternal DIREKOMENDASIKAN hanya tersedia secara eksternal menggunakan Service/API Gateway yang diperketat;
2. Autentikasi:
   1. Kekuatan mekanisme autentikasi yang digunakan HARUS memadai untuk kebutuhan proteksi;
   2. *Shared secret* yang digunakan untuk autentikasi layanan (misalnya kunci API atau OAuth 2.0 *Client Secret*) HARUS memiliki karakteristik berikut:
3. Panjang minimal 32 karakter (= 256 bit),
4. Acak kriptografis (terdiri dari alfanumerik dan karakter khusus),
5. Disimpan dengan aman sesuai dengan persyaratan yang ditentukan,
6. Ditransmisikan di luar URL (misalnya melalui HTTP POST atau header HTTP).
   1. Kredensial autentikasi HARUS unik untuk sistem dan lingkungan yang berbeda (misalnya pengujian, produksi);
7. Token akses:
   1. Token akses hanya berlaku:
8. *Short-lived,*
9. Lingkup terbatas,
10. Ditransmisikan hanya melalui HTTPS atau saluran yang aman,
11. Dikeluarkan oleh *trusted and hardened server* (misalnya OIDC *Identity Server* atau OAuth 2.0 *Authorization Server*),
12. Dibuat dan divalidasi menggunakan *mature API.*
13. Persyaratan OAuth 2.0/OICD:
    1. Only 3-Legged: Pemberian Kode Otorisasi dengan PKCE untuk kedua klien publik (misalnya *single-page application* (SPA) DIREKOMENDASIKAN digunakan untuk publik dan rahasia klien (misalnya aplikasi web sisi server),
    2. Only 3-Legged: Untuk perlindungan CSRF, Alur Pemberian Kode Otorisasi HARUS menggunakan parameter “state”, PKCE, atau “OICD Nounces”,
    3. Only 3-Legged: Klien HARUS mendaftar (dan divalidasi menggunakan) full redirect URI seperti yang dijelaskan dalam RFC 6819 Bagian 5.2.3.5 (*no pattern matching*).
    4. Akses layanan-ke-layanan DIREKOMENDASIKAN dilindungi menggunakan *Client Credential* *Grant.*
    5. HTTPS HARUS digunakan untuk semua komunikasi.
14. API UI Web:
    1. HARUS menerapkan persyaratan keamanan yang sama untuk akses ke sumber daya eksternal sebagai UI web terkait (otentikasi, validasi, dll.),
    2. HARUS tidak mengandung kode skrip (misalnya JavaScript) yang dapat dieksekusi secara langsung,
    3. HARUS berisi perlindungan CSRF jika bekerja dalam konteks pengguna.
15. *Cross-Domain Access:*
    1. Permintaan lintas domain HARUS dilakukan menggunakan mekanisme yang aman seperti *Cross-Origin Resource Sharing* (CORS) dan kebijakan yang membatasi (misalnya dibatasi sehingga *host* atau domain tertentu atau dengan pengiriman kredensial dinonaktifkan),
    2. *Header* asal permintaan lintas domain HARUS diotorisasi di sisi server.
16. Layanan DIREKOMENDASIKAN menerapkan batasan yang membatasi (permintaan maks. per klien atau interval waktu).
17. WebSockets HARUS mentransfer semua data rahasia dengan skema wss:// dan otorisasi sisi server tambahan dari header asal sumber.
18. Melakukan konfigurasi layanan web;
19. Memverifikasi uniform resource identifier API tidak menampilkan informasi yang berpotensi sebagai celah keamanan;
20. Membuat keputusan otorisasi;
21. Menampilkan metode RESTful hypertext transfer protocol apabila input pengguna dinyatakan valid;
22. Menggunakan validasi skema dan verifikasi sebelum menerima input;
23. Menggunakan metode pelindungan layanan berbasis web; dan
24. Menerapkan kontrol antiotomatisasi;

## Keamanan Konfigurasi

1. HARUS mengonfigurasi server sesuai rekomendasi server aplikasi dan kerangka kerja aplikasi yang digunakan;
2. HARUS mendokumentasi, menyalin konfigurasi, dan semua dependensi;
3. Menghapus fitur, dokumentasi, sampel, dan konfigurasi yang tidak diperlukan;
4. Memvalidasi integritas aset jika aset aplikasi diakses secara eksternal; dan
5. Menggunakan respons aplikasi dan konten yang aman
6. External services SHOULD only be made available externally using a hardened Service/API Gateway.

# Appendix A: Requirements for HTTP Security Header

Modern browsers support several protection mechanisms that can be activated and configured by a web application via HTTP response headers. The table below describes related requirements and recommendations for external web applications in production:

Headers that are centrally included:

|  |  |  |  |
| --- | --- | --- | --- |
| Response Header | General Requirement | For Web UIs? | For APIs? |
| Content-Type | ...; charset=utf-8 | Yes | Yes |
| Strict-Transport-Security[[1]](#footnote-1) | max-age=10886400; includeSubDomains; preload | Yes (if HTTPS) | Yes |
| X-XSS-Protection[[2]](#footnote-2) | 1; mode=block | Yes | No (but does no harm if yes) |
| X-Frame-Options | SAMEORIGIN | Yes | No |
| Referrer Policy | same-origin | Yes | No (but does no harm if yes) |
| X-Content-Type-Options[[3]](#footnote-3) | nosniff | Yes | No (but does no harm if yes) |

**Headers that must be set within the Web application:**

|  |  |  |  |
| --- | --- | --- | --- |
| Response Header | General Requirement | For Web UIs? | For APIs? |
| Set-Cookie | … ;httpOnly; secure; SameSite | |  | | --- | | Yes, when they transfer of confidential data in cookies | | Yes, when they transfer of confidential data in cookies |
| Cache-Control | no-cache, no-store | Whenever confidential data is transmitted. | Whenever confidential data is transmitted. |
| Pragma | no-cache |  |
| Expires | -1 |  |
| Content-Security-Policy[[4]](#footnote-4) | object-src 'none'; script-src ‘self’ [URL1] [URL2]; style-src ‘self’ unsafe-inline; object-src ‘self‘;base-uri 'none'; | General recommendation for new for all Web UIs. Not required for APIs. | No |
| object-src 'none'; script-src 'nonce-{random}' 'unsafe-inline' 'unsafe-eval' 'strict-dynamic' https: http:; base-uri 'none'; report-uri https://your-report-collector.example.com/ | Recommendation for new Web UIs that have to use inline script blocks (e.g. if integrated by a JS framework).  Do not use this setting if you do not have to since it disables CSP protection for older browsers! | No (but does no harm if yes) |
| Content-Disposition | attachment; filename=<filename> | Web UIs  at which users can download files that are potentially untrusted. | No (but does no harm if yes) |
| X-Download-Options | noopen | No (but does no harm if yes) |

**Caution: Settings these headers may have implications on the proper functionality of a web application. Therefore, activating a new header SHOULD always be combined with comprehensive functional tests.**

1. HTTP Strict Transport Security (HSTS) forces users of a web site to exclusively access it via HTTPS for a defined amount of time. Before using this header, you should ensure that all requests to this host (and to all subdomains when using „includeSubDomains” attribute) can be executed using HTTPS. This header prevents certain man-in-the-middle attacks. [↑](#footnote-ref-1)
2. Additional Cross-site Scripting prevention (IE only). [↑](#footnote-ref-2)
3. Deactivation of MIME sniffing in browsers that are used to identify MIME types but also to execute certain Cross-site Scripting attacks. [↑](#footnote-ref-3)
4. A Content Security Policy (CSP) provides an additional but very effective client-side protection against many common client-dite attacks such as Cross-site Scripting (XSS). Applications must explicitly support it which often requires a CSP compliant MVC framework. [↑](#footnote-ref-4)