

SPICKZETTEL · TOMORIS TOOLBOX

Security-Header-Spickzettel: empfohlene Werte und fertige Konfiguration

HTTP-Sicherheits-Header mit Zweck, empfohlenem Wert und Konfiguration für
nginx, Apache, IIS und Next.js

STAND

22.10.2026

UMFANG

PDF, 8 Seiten

ONLINE

tomoris.com/downloads/security-header-spickzettel

AUF EINEN BLICK

9

Header mit empfohlenem Startwert

4

fertige Konfigurationen: nginx,
Apache, IIS, Next.js

25

Quellen: OWASP, MDN, W3C und
Hersteller

FÜR WEN

- Wer Webserver mit nginx, Apache oder IIS betreibt
- Wer Websites und Webanwendungen entwickelt, etwa mit Next.js
- IT-Verantwortliche, die Befunde aus Prüfberichten oder Pentests umsetzen lassen

WURUM ES GEHT

Security-Header sind Anweisungen Ihres Webserver an den Browser. Sie kosten nichts und schließen ganze Angriffsklassen aus: Mitlesen über unverschlüsseltes HTTP, Clickjacking, eingeschleuste Skripte. Dieser Spickzettel nennt für jeden Header den empfohlenen Startwert, erklärt seinen Zweck und liefert die fertige Konfiguration für nginx, Apache, IIS und Next.js. Die Werte entsprechen den Empfehlungen des kostenlosen Security-Header-Checks von TOMORIS.

Dieser Spickzettel ersetzt keine Prüfung Ihrer Anwendung. Testen Sie Änderungen an Headern zuerst auf einer Testumgebung: Eine zu strenge Content-Security-Policy kann Funktionen Ihrer Website blockieren.

INHALT

01	Die Header im Überblick	2
02	In vier Schritten einführen	3
03	Was weg sollte: Versionsangaben	4
04	Konfiguration für nginx	5
05	Konfiguration für Apache	6
06	Konfiguration für IIS	6
07	Konfiguration für Next.js	7
08	Quellen	8

01

Die Header im Überblick

Neun Header mit empfohlenem Startwert, Zweck und Hinweis zur Einführung, dazu die strengeren Werte von OWASP.

Empfohlene Security-Header (Startwerte wie im Security-Header-Check von TOMORIS)

Strict-Transport-Security

`max-age=31536000; includeSubDomains`

Der Browser ruft die Website für die Dauer von max-age (hier ein Jahr) nur noch über HTTPS auf. Nur über HTTPS senden, über HTTP ignoriert der Browser den Header. includeSubDomains erst setzen, wenn alle Subdomains HTTPS können. OWASP schlägt zwei Jahre vor.

Content-Security-Policy

`default-src 'self'; script-src 'self';
object-src 'none'; base-uri 'self';
frame-ancestors 'self'; form-action
'self'; upgrade-insecure-requests`

Legt fest, woher Skripte und andere Inhalte laden dürfen. Der wichtigste Schutz gegen Cross-Site-Scripting. Zuerst als **Content-Security-Policy-Report-Only** einführen: meldet Verstöße, blockiert nichts. Inline-Skripte über Nonces oder Hashes freigeben statt 'unsafe-inline'.

X-Frame-Options

`SAMEORIGIN`

Schutz vor Clickjacking für ältere Browser: Nur die eigene Website darf die Seite einbetten. Moderner ist `frame-ancestors` in der CSP; ist es gesetzt, wird X-Frame-Options laut CSP-Standard des W3C ignoriert. ALLOW-FROM ist veraltet und wird ignoriert. Streng: DENY bzw. frame-ancestors 'none' (OWASP).

X-Content-Type-Options

`nosniff`

Der Browser errät keine Dateitypen und blockiert Skripte und Stylesheets mit falschem MIME-Typ. Dateien müssen mit korrektem Content-Type ausgeliefert werden, sonst blockiert der Browser sie.

Referrer-Policy

`strict-origin-when-cross-origin`

Fremde Websites erfahren höchstens Ihre Domain, nicht die aufgerufene Adresse samt Parametern. Ist der Standard aktueller Browser, wenn keine Policy gesetzt ist; der Header macht ihn verbindlich. Streng: no-referrer (OWASP).

Permissions-Policy

`camera=(), microphone=(), geo-location=(), payment=(), usb=(), browsing-topics=()`

Sperrt Browserfunktionen, die die Website nicht braucht, auch für eingebettete Inhalte. Nur sperren, was die Website nicht nutzt. OWASP sperrt weitere Funktionen. browsing-topics ist laut MDN veraltet und kann entfallen.

Cross-Origin-Opener-Policy

`same-origin-allow-popups`

Fremde Fenster, die Ihre Seite öffnen, erhalten keinen Zugriff auf sie. Streng: same-origin (OWASP). same-origin-allow-popups passt, wenn Anmelde- oder Zahlungsfenster fremder Anbieter geöffnet werden.

Cross-Origin-Resource-Policy

`same-site`

Fremde Websites können Ihre Ressourcen nicht einbinden. Streng: same-origin (OWASP). Für Inhalte, die bewusst überall eingebunden werden sollen: cross-origin.

X-XSS-Protection

`0`

Weglassen oder `0`. Veralteter Filter älterer Browser, der in manchen Fällen selbst Lücken öffnete. Den Schutz übernimmt die Content-Security-Policy (MDN, OWASP).

Die empfohlenen Werte sind die Startwerte des Security-Header-Checks von TOMORIS. Wo OWASP strengere Werte empfiehlt, steht das jeweils beim Header. Testen Sie jede Änderung zuerst auf einer Testumgebung.

02

In vier Schritten einführen

Bestand messen, unkritische Header setzen, CSP im Beobachtungsmodus testen, dann durchsetzen.

1

Bestand messen

Prüfen Sie, welche Header Ihre Website heute sendet, zum Beispiel mit dem kostenlosen Security-Header-Check von TOMORIS unter tomoris.com/tools.

2

Einfache Header zuerst

Beginnen Sie mit `X-Content-Type-Options`, `Referrer-Policy`, `X-Frame-Options`, der Umleitung von HTTP auf HTTPS und dem Entfernen von Versionsangaben. Prüfen Sie vorher, ob fremde Websites Ihre Seiten absichtlich einbetten und ob alle Dateien mit korrektem Content-Type ausgeliefert werden.

3

HSTS und CSP stufenweise

HSTS mit kurzer Laufzeit beginnen und schrittweise erhöhen (hstspreload.org nennt die Stufen 5 Minuten, 1 Woche und 1 Monat), dann ein Jahr. Die CSP zuerst als `Content-Security-Policy-Report-Only` ausliefern und die gemeldeten Verstöße auswerten.

4

Durchsetzen und wiederholen

Dieselbe Richtlinie als `Content-Security-Policy` setzen und nach jeder größeren Änderung an der Website erneut prüfen.

03 —

Was weg sollte: Versionsangaben

Server-Versionen und X-Powered-By verraten Angreifern, nach welchen bekannten Lücken sie suchen müssen.

Header wie `Server: nginx/1.24.0` oder `X-Powered-By: PHP/8.1.2` nennen Software und Version. OWASP führt sie in seiner Liste der Header, die nach Möglichkeit entfernt werden sollten, damit keine technischen Details preisgegeben werden.

- `nginx: server_tokens off`; entfernt die Versionsnummer aus dem Server-Header und von Fehlerseiten. Der Header selbst bleibt.
- Apache: `ServerTokens Prod` (nur in der Serverkonfiguration) sendet nur noch `Server: Apache; ServerSignature Off` ist bereits der Standard.
- PHP: `expose_php = Off` in der `php.ini` entfernt den Header X-Powered-By mit der PHP-Version.
- IIS: `<remove name="X-Powered-By" />` in den `customHeaders` und `removeServerHeader="true"` in request-Filtering (ab IIS 10.0 Version 1709).
- Next.js: `poweredByHeader: false` in der `next.config`.

Das Verbergen von Versionen ersetzt keine Updates. Auch die Apache-Dokumentation nennt Sicherheit durch Verschleierung einen Mythos. Es nimmt Angreifern aber eine bequeme Abkürzung.

04

Konfiguration für nginx

add_header mit always und die Falle bei der Vererbung in location-Blöcken.

nginx: server-Block

```
server {
    listen 443 ssl;
    server_name example.com;
    server_tokens off;

    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
    add_header Content-Security-Policy "default-src 'self'; script-src 'self'; object-src 'none'; base-uri
    'self'; frame-ancestors 'self'; form-action 'self'; upgrade-insecure-requests" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "strict-origin-when-cross-origin" always;
    add_header Permissions-Policy "camera=(), microphone=(), geolocation=(), payment=(), usb=(),
    browsing-topics=()" always;
    add_header Cross-Origin-Opener-Policy "same-origin-allow-popups" always;
    add_header Cross-Origin-Resource-Policy "same-site" always;
}

# HTTP dauerhaft auf HTTPS umleiten
server {
    listen 80;
    server_name example.com;
    return 301 https://$host$request_uri;
}
```

- Ohne **always** setzt nginx die Header nur bei den Statuscodes 200, 201, 204, 206, 301, 302, 303, 304, 307 und 308, also nicht auf Fehlerseiten.
- Vererbung: add_header-Direktiven gelten nur dann aus der übergeordneten Ebene, wenn auf der aktuellen Ebene kein eigenes add_header steht. Ein einziges add_header in einem location-Block lässt dort alle Security-Header der server-Ebene wegfallen.
- Ab nginx 1.29.3 (stabil ab 1.30) übernimmt **add_header_inherit merge**; die Header der übergeordneten Ebene zusätzlich.

05 —

Konfiguration für Apache

Header always set mit mod_headers, Versionsangaben und Umleitung auf HTTPS.

Apache httpd 2.4

```
# VirtualHost für HTTPS oder .htaccess (mod_headers)
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
Header always set Content-Security-Policy "default-src 'self'; script-src 'self'; object-src 'none';
  base-uri 'self'; frame-ancestors 'self'; form-action 'self'; upgrade-insecure-requests"
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-Content-Type-Options "nosniff"
Header always set Referrer-Policy "strict-origin-when-cross-origin"
Header always set Permissions-Policy "camera=(), microphone=(), geolocation=(), payment=(), usb=(),
  browsing-topics=()"
Header always set Cross-Origin-Opener-Policy "same-origin-allow-popups"
Header always set Cross-Origin-Resource-Policy "same-site"
Header unset X-Powered-By

# nur in der Serverkonfiguration (httpd.conf), nicht in .htaccess
ServerTokens Prod
ServerSignature Off

# VirtualHost für Port 80: dauerhaft auf HTTPS umleiten
Redirect permanent / https://example.com/
```

- **always** setzt die Header auch bei Fehlerantworten und nach internen Weiterleitungen, etwa durch ErrorDocument.
- In einer .htaccess wirkt **Header** nur, wenn die Serverkonfiguration dort **AllowOverride FileInfo** erlaubt.
- **Redirect permanent** antwortet mit Status 301.

06 —

Konfiguration für IIS

customHeaders in der web.config, Server-Header entfernen und natives HSTS ab IIS 10.0 Version 1709.

IIS: web.config

```
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
        <remove name="X-Powered-By" />
        <add name="Strict-Transport-Security" value="max-age=31536000; includeSubDomains" />
        <add name="Content-Security-Policy" value="default-src 'self'; script-src 'self'; object-src
          'none'; base-uri 'self'; frame-ancestors 'self'; form-action 'self'; upgrade-insecure-requests"
          />
        <add name="X-Frame-Options" value="SAMEORIGIN" />
        <add name="X-Content-Type-Options" value="nosniff" />
        <add name="Referrer-Policy" value="strict-origin-when-cross-origin" />
        <add name="Permissions-Policy" value="camera=(), microphone=(), geolocation=(), payment=(),
          usb=(), browsing-topics=()" />
        <add name="Cross-Origin-Opener-Policy" value="same-origin-allow-popups" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

```
<add name="Cross-Origin-Resource-Policy" value="same-site" />
</customHeaders>
</httpProtocol>
<security>
  <requestFiltering removeServerHeader="true" />
</security>
</system.webServer>
</configuration>
```

- **removeServerHeader** gibt es ab IIS 10.0 und wirkt erst ab Windows Server bzw. Windows 10, jeweils Version 1709.
- Für HSTS samt Umleitung von HTTP auf HTTPS kennt IIS ab 10.0 Version 1709 das Element **<hsts enabled="true" max-age="31536000" includeSubDomains="true" redirectHttpToHttps="true" />** je Site in der applicationHost.config, nicht in der web.config. Lassen Sie dann den customHeader für HSTS weg, damit der Header nicht doppelt gesendet wird.

07

Konfiguration für Next.js

headers() in der next.config, x-powered-by abschalten und was eine strenge CSP mit Nonces kostet.

Next.js: next.config.ts

```
import type {NextConfig} from 'next';

// Ohne Nonces braucht Next.js 'unsafe-inline' für Skripte und Styles
// (Next.js-Leitfaden „Content Security Policy“, Abschnitt „Without Nonces“)
const securityHeaders=[
  {key:'Strict-Transport-Security',value:"max-age=31536000; includeSubDomains"},
  {key:'Content-Security-Policy',value:"default-src 'self'; script-src 'self' 'unsafe-inline'; style-src 'self' 'unsafe-inline'; img-src 'self' blob: data:; font-src 'self'; object-src 'none'; base-uri 'self'; frame-ancestors 'self'; form-action 'self'; upgrade-insecure-requests"},
  {key:'X-Frame-Options',value:"SAMEORIGIN"},
  {key:'X-Content-Type-Options',value:"nosniff"},
  {key:'Referrer-Policy',value:"strict-origin-when-cross-origin"},
  {key:'Permissions-Policy',value:"camera=(), microphone=(), geolocation=(), payment=(), usb=(), browsing-topics=()"},
  {key:'Cross-Origin-Opener-Policy',value:"same-origin-allow-popups"},
  {key:'Cross-Origin-Resource-Policy',value:"same-site"},
];

const nextConfig:NextConfig={
  poweredByHeader:false,
  async headers(){
    return [{source:'/:path*',headers:securityHeaders}];
  },
};

export default nextConfig;
```

- Ohne Nonces nennt der Next.js-Leitfaden **'unsafe-inline'** für Skripte und Styles. Der Security-Header-Check wertet das als Hinweis, weil die CSP damit eingeschleuste Inline-Skripte nicht mehr blockiert.

- Strenger geht es mit Nonces: Sie werden im Proxy erzeugt (`proxy.ts`, bis Next.js 15 Middleware). Dann müssen alle Seiten dynamisch gerendert werden; statische Optimierung und ISR entfallen.

NÄCHSTER SCHRITT

Selbst prüfen oder gemeinsam umsetzen

KOSTENLOS PRÜFEN

Security-Header-Check

tomoris.com/tools/security-header-check

UMSETZUNG BESPRECHEN

Kontakt aufnehmen

tomoris.com/contact

contact@tomoris.com

08

Quellen

Alle Aussagen sind belegt. Gesetzestexte gelten in der jeweils geltenden Fassung.

- 1 OWASP Secure Headers Project: Best Practices und Liste empfohlener Header
[Werte aus ci/headers_add.json](https://owasp.org/www-project-secure-headers), Stand 18.10.2026
owasp.org/www-project-secure-headers
- 2 OWASP Secure Headers Project: zu entfernende Header (`headers_remove.json`)
github.com/OWASP/secure-headers-project/blob/master/ci/headers...
- 3 MDN: Strict-Transport-Security
developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Strict...
- 4 MDN: Content-Security-Policy und Content-Security-Policy-Report-Only
developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Conten...
- 5 MDN: X-Frame-Options
developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/X-Fram...
- 6 W3C Content Security Policy Level 3, Abschnitt 6.4.2.2 (frame-ancestors und X-Frame-Options)
www.w3.org/TR/CSP3
- 7 MDN: X-Content-Type-Options
developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/X-Cont...
- 8 MDN: Referrer-Policy
developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Referr...
- 9 MDN: Permissions-Policy (und browsing-topics, veraltet)
developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Permis...
- 10 MDN: Cross-Origin-Opener-Policy
[developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Cross...](https://developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Cross-...)
- 11 MDN: Cross-Origin-Resource-Policy
[developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Cross...](https://developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/Cross-...)
- 12 MDN: X-XSS-Protection
[developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/X-XSS...](https://developer.mozilla.org/de/docs/Web/HTTP/Reference/Headers/X-XSS-...)
- 13 nginx: Modul ngx_http_headers_module (add_header, always, add_header_inherit)
nginx.org/en/docs/http/ngx_http_headers_module.html
- 14 nginx: server_tokens
nginx.org/en/docs/http/ngx_http_core_module.html#server_tokens
- 15 Apache httpd 2.4: mod_headers
httpd.apache.org/docs/2.4/mod/mod_headers.html
- 16 Apache httpd 2.4: ServerTokens und ServerSignature
httpd.apache.org/docs/2.4/mod/core.html#servertokens
- 17 Apache httpd 2.4: mod_alias (Redirect permanent)
httpd.apache.org/docs/2.4/mod/mod_alias.html
- 18 PHP-Handbuch: expose_php
www.php.net/manual/en/ini.core.php
- 19 Microsoft Learn: IIS customHeaders
learn.microsoft.com/iis/configuration/system.webserver/httpprot...
- 20 Microsoft Learn: IIS requestFiltering (removeServerHeader)
learn.microsoft.com/iis/configuration/system.webserver/security...
- 21 Microsoft Learn: IIS HSTS je Site
learn.microsoft.com/iis/configuration/system.applicationhost/si...
- 22 hstspreload.org: Voraussetzungen und schrittweise Einführung von HSTS
hstspreload.org
- 23 Next.js: headers in der next.config
nextjs.org/docs/app/api-reference/config/next-config-js/headers
- 24 Next.js: poweredByHeader
nextjs.org/docs/app/api-reference/config/next-config-js/powered...
- 25 Next.js: Leitfaden Content Security Policy (Nonces, Proxy, Without Nonces)
nextjs.org/docs/app/guides/content-security-policy