decareto

Monitoring Report

Website: www.bundestag.de/

Zeitpunkt des Scans: 04.07.2023 17:34

Risikobewertung

A	1	1	
Risiko-Score	Externe Dienste gefunden	Cookies gefunden	Datenschutzerklärung gefunden

Die Website läuft im Netzwerk von **Babiel GmbH**. Der Serverstandort ist Deutschland.

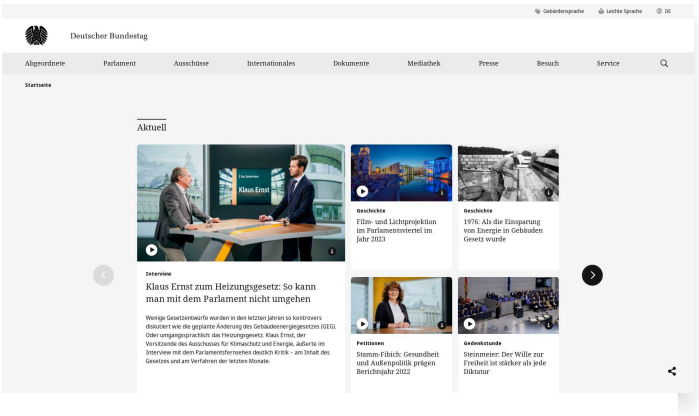
Die Prüfung der Server-Sicherheit ergab keine Auffälligkeiten.

Wir konnten keine nicht notwendigen Cookies identifizieren, die ohne Einwilligung des Benutzers gesetzt werden.
Achtung: Wir untersuchen nicht, ob das Einholen der Einwilligung DSGVO-konform erfolgt!

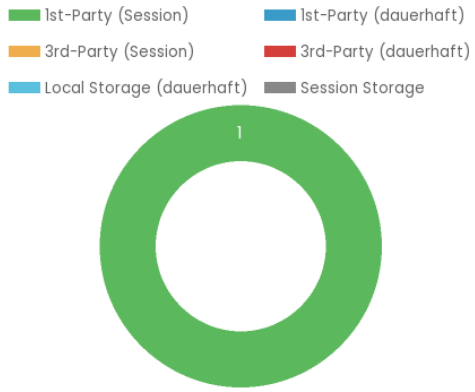
Wir konnten keine nicht notwendigen Dienste identifizieren, die ohne Einwilligung des Benutzers geladen werden.
Achtung: Wir untersuchen nicht, ob das Einholen der Einwilligung DSGVO-konform erfolgt!

Bitte beachten Sie, dass trotz aller Sorgfalt bei der Untersuchung nicht ausgeschlossen werden kann, dass die Website Datenschutzschwachstellen aufweist, die in diesem Report nicht aufgezeigt werden. Wir können keine Haftung für die Vollständigkeit dieses Reports übernehmen.

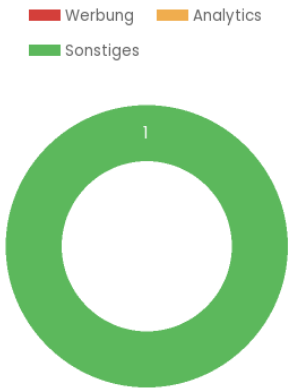
Ansicht Startseite



Cookies und Web-Speicher



Cookie-Typ



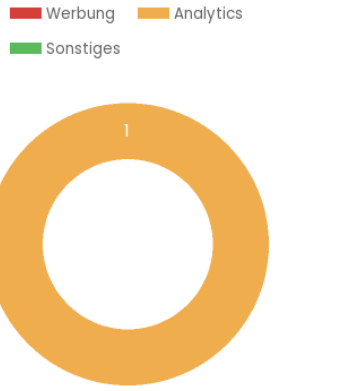
Verwendung

Einwilligung zum Setzen

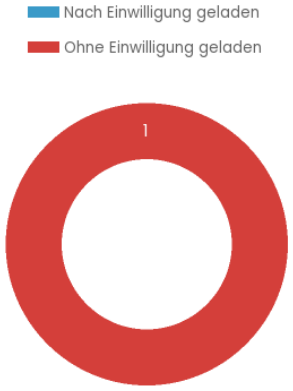
Cookies und Web-Speicher Übersicht

Name	Typ	Speicherdauer (Tage)	Domain	Quelle	Datentransfer	Zweck	Einwilligung erteilt
SERVERID	1st-Party (Session)	0	bundestag.de	Webserver		Funktional	Nein

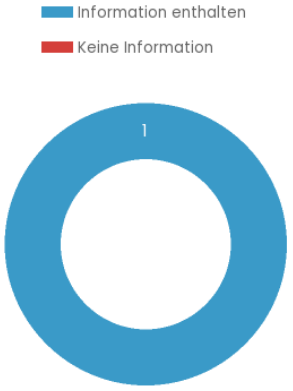
Externe Dienste



Verwendung



Einwilligung zum Laden



Information in Datenschutzerklärung

Externe Dienste Übersicht

Name	Domain	Quelle	Datentransfer	Zweck	Einwilligung erteilt	Information in Datenschutzerklärung
Matomo Auf der Startseite gefunden	|piwik.js	Matomo		Analytics	Nein	Ja

TLS/SSL-Verschlüsselung und Sicherheit des Webservers

- Das Zertifikat enthält korrekte und vollständige Informationen [1]
- Das Zertifikat ist zeitlich gültig bis 10.09.2023
- Das Zertifikat wird akzeptiert auf allen gängigen Plattformen (Apple, Android, Oracle/Java, Microsoft/Windows, Mozilla/Firefox) [2]

- Der Server ist geschützt gegen die verbreitetsten TLS/SSL-Angriffe [3]
- Der Webserver akzeptiert keine veralteten und unsicheren TLS/SSL-Protokolle. [4]
- Die aktuellen Protokolle TLS 1.2 bzw. TLS 1.3 werden akzeptiert [5]
- Für den Webserver sind Umleitungen von HTTP zu HTTPS korrekt konfiguriert [6]
- Der Webserver setzt zwar HTTP-Header für Content-Security-Policy, diese ist aber nicht ausreichend sicher konfiguriert [7]
- Für den Webserver ist HTTP Strict Transport Security ausreichend sicher konfiguriert [8]
- Der Webserver setzt keine Header für eine Referrer-Policy [9]
- Der Webserver setzt einen *X-Content-Type-Options*-Header [10]
- Der Webserver setzt einen sicheren *X-Frame-Options*-Header [11]

Zur Ermittlung der Sicherheit des Webservers verwenden wir Mozilla Observatory, für das komplette Scan-Ergebnis klicken Sie bitte hier.

Erläuterungen und Handlungsempfehlungen

[1] Wir untersuchen das TLS/SSL-Zertifikat darauf, ob der Server-Name im Zertifikat mit dem tatsächlichen Servernamen übereinstimmt, und ob das Zertifikat von einer vertrauenswürdigen Quelle stammt. Wenn eins von beiden nicht gegeben ist, zeigt ein Web-Browser normalerweise an, dass die Verbindung nicht sicher ist, weil in diesen Fällen sog. "Man-in-the-middle-Angriffe" möglich sind. Außerdem prüfen wir, ob die "Intermediate-Zertifikate" auf dem Server enthalten sind, die die Vertrauenswürdigkeit des Ausstellers nachweisen. Wenn diese fehlen, dann zeigen ältere Web-Browser möglicherweise Fehler an. Die Prüfungen zeigten keine Probleme.

[2] Die Zertifizierungsstelle, über die das Zertifikat des Webservers erworben wurden, muss von den großen Plattformen (Apple, Android, Oracle/Java, Microsoft/Windows, Mozilla/Firefox) als vertrauenswürdig eingestuft und in deren "Trust Store" aufgenommen worden sein. Wenn das nicht der Fall ist, dann stufen die Geräte dieser Plattformen das Zertifikat als nicht gültig ein. Im Fall dieses Webservers wird das Zertifikat von allen Plattformen als vertrauenswürdig eingestuft.

[3] Wir untersuchen den Server auf die Schwachstellen "Heartbleed", "CRIME" und "Downgrade". Alle drei stehen in Zusammenhang mit veralteter Systemsoftware oder dem Akzeptieren veralteter Verschlüsselungsprotokolle. Wir konnten bei dem Server diese Schwachstellen nicht feststellen.

[4] Veraltete TLS/SSL-Protokolle bieten keine sichere Verschlüsselung mehr, so dass Daten für Angreifer sichtbar sein können. Insbesondere die sehr alten Protokolle SSL 2.0 und SSL 3.0 sollten auf keinen Fall mehr eingesetzt werden, aber auch TLS 1.0 und TLS 1.1 sind nicht mehr sicher genug. Der Webserver ist korrekt konfiguriert und akzeptiert diese Protokolle nicht.

[5] Der Webserver sollte für ausreichende Sicherheit die neuen TLS/SSL-Protokolle TLS 1.3 und ggfs. TLS 1.2 unterstützen. Der Webserver ist korrekt konfiguriert und unterstützt diese.

[6] Alle Aufrufe zu unverschlüsselten URLs werden so umgeleitet, dass der Aufruf verschlüsselt erfolgt.
Hintergrund: Der Server sollte so konfiguriert sein, dass unverschlüsselte Aufrufe sofort auf die entsprechende HTTPS-URL umgeleitet werden. Andernfalls könnte ein Benutzer bspw. verleitet werden, Formulardaten unverschlüsselt zu übertragen. Sobald die Umleitung stattgefunden hat, sollte der Browser per HSTS angewiesen werden, in Zukunft nur noch die verschlüsselte Verbindung zu benutzen. Dabei sollte die Umleitung nicht zu einer anderen Domain/Host führen (das würde HSTS aushebeln), sondern unmittelbar zur gleichen URL, aber mit HTTPS.

[7] Die Content Security Policy (CSP) ist nicht ausreichend restriktiv. Sie enthält die Direktiven *unsafe-eval* oder *data* innerhalb der Direktive *script-src*, zu weit gefasste Einschränkungen wie *https:* innerhalb der Direktive *object-src*, oder sie schränkt die Quellen für *object-src* or *script-src* nicht ein.
Hintergrund: Ein Content Security Policy (CSP)-HTTP-Header ist eine von mehreren möglichen Maßnahmen, um Websites gegen Angriffe durch Cross-Site-Scripting (XSS) zu schützen. Beim XSS injizieren Angreifer Javascript-Code in eine Seite (bspw. indem sie einen Blog-Kommentar schreiben, der Javascript-Code enthält). Wenn andere Besucher die Seite öffnen, wird das Javascript ausgeführt, was eine Vielzahl von Angriffsmöglichkeiten bietet, etwa das Auslesen und Versenden von Passwörtern während der Eingabe. Ein CSP-Header kann das verhindern, indem er das Ausführen von sog. "Inline-" Javascript grundsätzlich unterbindet, und nur Javascript von bestimmten Servern erlaubt, bzw. grundsätzlich das Laden von Ressourcen auf ausgewählte Server einschränkt. Bei der Einführung einer CSP muss möglicherweise der Anwendungscode angepasst werden, so ist u.A. der Einsatz des Google Tag

Managers nicht mehr ohne Weiteres möglich. Die Herausforderungen bei der Einführung einer CSP beschreiben dieser und dieser Artikel.

[8] Der Parameter *max-age*, der angibt, wie lange per HSTS verschlüsselte Verbindungen erzwungen werde sollen, ist korrekt auf mehr als 6 Monate konfiguriert. Ein Wert von ein bis zwei Jahren ist optimal.
Hintergrund: HTTP Strict Transport Security (HSTS) ist ein Sicherheitsmechanismus, bei dem der Server einem Browser mitteilt, dass für eine bestimmte Zeit ausschließlich verschlüsselte Verbindungen verwendet werden dürfen. Bei so genannten Man-in-the-Middle Angriffen versucht ein Angreifer, den Aufbau einer verschlüsselten Verbindung zu verhindern, ohne dass der Benutzer etwas davon merkt. Der Angreifer kann dann unbemerkt alle übermittelten Daten mitlesen. Mit HSTS soll bereits am Beginn der Verbindung eine HTTPS Verschlüsselung erzwungen und damit die Gefahr solcher Angriffe minimiert werden. Ein Webserver sollte für optimalen Schutz immer HSTS in Verbindung mit einer HTTPS-Umleitung verwenden.

[9] Es ist zwar kein sehr großes Sicherheitsrisiko, keine Referrer-Policy festzulegen, aber auch nicht ideal. Die Browser verwenden dann eine eigene Policy, die möglicherweise etwas unsicherer ist als die empfohlene, das Verhalten ist aber auf jeden Fall unvorhersehbar.
Hintergrund: Der Referrer ist ein HTTP-Header, der bei einem Aufruf (auch an externe Ressourcen) die vorherige bzw. die aktuelle URL mitteilt. Da die URL sensitive Informationen enthalten kann, ist dies ein potentielles Sicherheitsrisiko. Eine Referrer-Policy legt deswegen fest, bei welchen Aufrufen dieser Header welchen Teil der URL enthält (oder leer ist). Als Best Practice gilt es, dass der Header die Policy *strict-origin-when-cross-origin* festlegt, dabei enthält dann der Referrer nur den eigenen Servernamen, wenn ein Aufruf an fremde Server stattfindet. Detaillierte Hinweise finden Sie hier.

[10] Es wurde ein *X-Content-Type-Options*-Header mit dem korrekten Wert *nosniff* gefunden.
Hintergrund: Der *X-Content-Type-Options*-Header mit dem Inhalt *nosniff* dient dazu, Cross-Site-Scripting-Attacken abzuwehren. Diese können auftreten, weil manche Browser (bspw. Internet Explorer) ein sog. "Content-Sniffing" durchführen. Dabei versucht der Browser selber herauszufinden, welchen Inhaltstyp eine Ressource hat, wenn der Content-Type-Header fehlt. Das ermöglicht es einem Angreifer, Javascript in eine Seite zu injizieren, etwa wenn er die Möglichkeit hat, selber Inhalte in ein Forum o.Ä. hochzuladen. Wenn andere Besucher die Seite öffnen, wird das Javascript ausgeführt, was eine Vielzahl von Angriffsmöglichkeiten bietet, etwa das Auslesen und Versenden von Passwörtern während der Eingabe. Es ist deshalb sinnvoll, diesen Header zu setzen.

[11] Es wurde ein *X-Frame-Options*-Header mit einem sicheren Wert *SAMEORIGIN* oder *DENY* gefunden.
Hintergrund: Der *X-Frame-Options*-Header legt fest, ob die Website über ein iFrame auf einer anderen Website eingebettet werden kann. Letzteres kann ein Sicherheitsrisiko durch Clickjacking darstellen. Dabei werden Teile der eingebetteten Website durch Elemente des Angreifers überlagert, etwa um einem Besucher dazu zu bringen, auf scheinbar harmlose - aber tatsächlich gefährliche - Links zu klicken. Mit Hilfe des Headers kann derartiges Einbetten unterbunden werden. Alternativ ist der Content-Security-Policy-Header ebenfalls geeignet, ein Einbetten zu verhindern.