



Wie Sie Ihre Website durch das Setzen von Security Headern sicherer machen

Umgebung: Microsoft Internet Information Services (IIS) Manager

Anya Raider, Cyber Security Consultant

Inhalt

1.	Übersicht	3
2.	Website Hardening Maßnahmen	3
2.1	ASP.NET und Server Versionsoffenlegung verhindern	4
2.2	Strict-Transport-Security	5
2.3	X-Powered-By HTTP Header	6
2.4	X-Content-Type-Options	6
2.5	X-Frame-Options	7
2.6	X-XSS-Protection	8
2.7	Referrer-Policy	8
2.8	Update der TLS und SSL Version	9
2.9	Cache-Control	11
2.10	Insecure HTTP Method – TRACE and TRACK	11
2.11	Content Security Policy	12
2.12	Permissions Policy Header	17
2.13	Internal IP address disclosed	18
2.14	Cookie SameSite Attribute	18
2.15	Anti-CSRF Tokens	19
3.	Security-Header Analyse	23
	Über GBS	23



1. Übersicht

Es gibt viele Wege die eigene Website, die offen im Internet zugänglich ist, zu schützen. Der jeweilige Schutz hängt auch von der Infrastruktur ab, von der aus die Seite administriert wird. Diese Dokumentation konzentriert sich auf die Microsoft Internet Information Services (IIS), die in der Regel auf Servern betrieben werden. Während der IIS sämtliche Services bereitstellt, liegt der Fokus in dieser Anleitung jedoch auf den „HTTP Response Headers“, „Configuration Editor“ und „Request Filtering“ Services des Managers.

Das Ziel ist es, Ihnen dabei zu helfen, Ihre HTTP Response Headers zu „harden“, das bedeutet zu stärken und damit Ihre Website weniger verwundbar zu machen. Durch Änderungen im Configuration Editor und Anpassungen im Request Filtering wird der Schutz verstärkt.

Im Folgenden sehen Sie eine Auflistung an Schutzmaßnahmen, die Sie direkt in Ihrem IIS vornehmen können. Im letzten Teil dieses Dokumentes werden Ihnen hilfreiche Links zur Verfügung gestellt, mithilfe derer Sie sofort untersuchen können, welche Header schon bei Ihnen gesetzt sind und welche noch fehlen.

2. Website Hardening Maßnahmen

Bevor wir mit den einzelnen Headern anfangen, sollten wir verstehen, was Header überhaupt bedeuten. Sobald wir eine Website aufrufen, übermittelt der Server direkt seine Informationen an den Browser des Nutzers. Ein Beispiel anhand meines Firefox Browsers: Ich rufe google.com auf und gelange durch F12 in die Netzwerkanalyse. Dort klicke ich auf den ersten „GET“ Eintrag und schaue mir die Antwortkopfzeilen an:

```
? strict-transport-security: max-age=31536000
? x-content-type-options: nosniff
? x-frame-options: ALLOW-FROM https://www.google.com
  x-ua-compatible: IE=edge
? x-xss-protection: 0
```

Dies ist nur ein Ausschnitt von vielen anderen Headern. Die HTTP Header übermitteln Parameter und Argumente, die für die Übertragung wichtig sind. Beim Setzen eines inkompatiblen Headers kann sich beispielsweise das Erscheinungsbild der Website verändern, weil das Zusammenspiel mehrerer Plugins nicht mehr funktioniert. Die hier aufgelisteten Maßnahmen, sind bereits erprobt und werden von uns selbst angewendet. Header, wie die Content Security Policy, müssen mit Vorsicht genossen werden, da sie bei komplexer aufgebauten Seiten oder bei Seiten, die ältere Plugins verwenden, zu Störungen führen können.

2.1 ASP.NET und Server Versionsoffenlegung verhindern

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Pragma: no-cache
4 Content-Type: application/json; charset=utf-8
5 Expires: -1
6 Server: Microsoft-IIS/10.0
7 X-AspNet-Version: 4.0.30319
8 X-Powered-By: ASP.NET
9 Date: Mon, 15 Feb 2021 15:05:27 GMT
10 Connection: close
11 Content-Length: 46
```

Information:

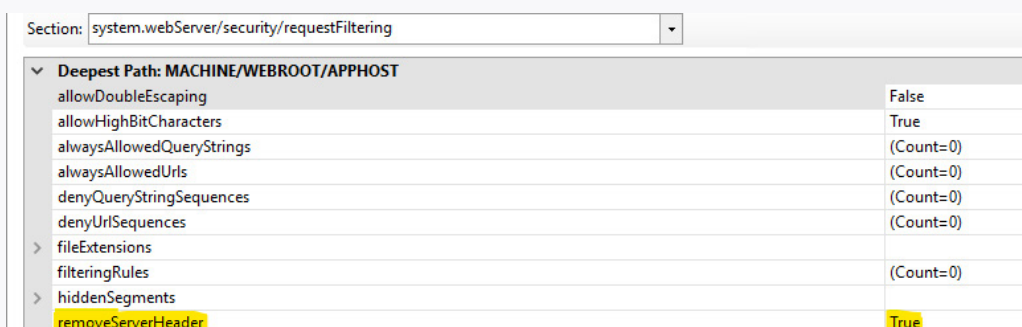
Die von der Webanwendung zurückgegebenen HTTP-Antworten enthalten einen Header namens X-AspNet-Version. Je mehr Informationen die Angreifer über Ihre Website haben, desto einfacher wird es für sie Sie zu treffen. Daher unterstützt man mit der folgenden Einstellung den Ansatz, die Offenlegung von Informationen so geringfügig wie möglich zu halten.

Lösung:

Wenden Sie die folgenden Änderungen an der Datei web.config an, um die Offenlegung der ASP.NET-Version zu verhindern. Dazu gehen Sie auf „Configuration Editor“ und ins Verzeichnis system.web/httpRuntime, dann auf enableVersionHeader und setzen den Wert auf "false":



1. Gehen Sie zu system.webServer/security/requestFiltering.
2. Setzen Sie removeServerHeader auf True.
3. Actions - Apply.

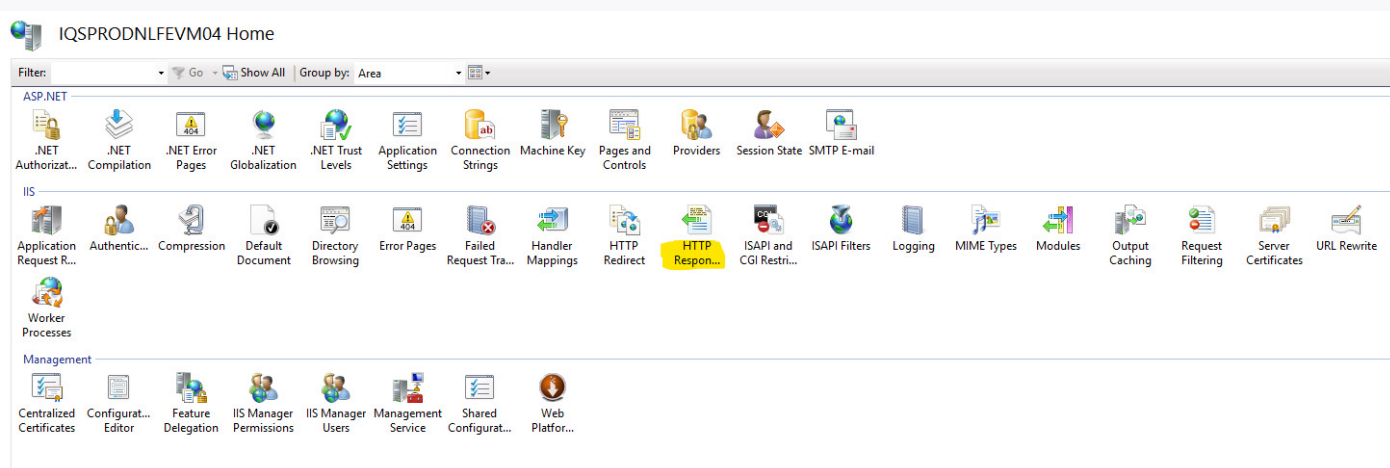


2.2 Strict-Transport-Security

Information:

Der Header informiert die Browser darüber, dass der Zugriff auf die Website nur über HTTPS erfolgen darf. Jeder Zugriffsversuch über HTTP wird automatisch in HTTPS umgewandelt. Wenn die in der Kopfzeile Strict-Transport-Security angegebene Ablaufzeit verstrichen ist, wird der nächste Versuch, die Website über HTTP zu laden, normal fortgesetzt, anstatt automatisch HTTPS zu verwenden. Sollte es notwendig sein, die strikte Transportsicherheit zu deaktivieren, führt die Einstellung von max-age auf 0 (über eine https-Verbindung) zum sofortigen Ablauf des Strict-Transport-Security-Headers und ermöglicht den Zugriff über http.

Um diese und die folgenden Header zu setzen, öffnen Sie Ihren IIS Manger. Über „HTTP Response Headers“ gelangen Sie dann zu dem relevanten Service.



Um einen neuen Header anzulegen, klicken Sie rechts unter „Actions“ auf „Add“ und fügen diesen Header hinzu:

Name:	<input type="text" value="Strict-Transport-Security"/>
Value:	<input type="text" value="max-age=31536000; includeSubDomains"/>

`max-age=<expire-time>`

Die Zeit, in Sekunden: Der Browser sollte sich merken, dass der Zugriff auf eine Website nur über HTTPS erfolgen darf. Jedes Mal, wenn der Strict-Transport-Security-Header an den Browser übermittelt wird, wird die Ablaufzeit für diese Website aktualisiert, so dass Websites diese Informationen aktualisieren und den Ablauf des Timeouts verhindern können.

`includeSubDomains`

Die Regel gilt auch für alle Subdomänen der Website.



2.3 X-Powered-By HTTP Header

Information:

Der X-Powered-By HTTP Header kann einfach über IIS entfernt werden. Standardmäßig ist er angelegt, weshalb man ihn dann einfach entfernen kann:

Lösung:

The screenshot shows the 'HTTP Response Headers' configuration window in IIS. The 'X-Powered-By' header is selected, and a context menu is open with 'Remove' highlighted. To the right, a confirmation dialog box asks 'Are you sure that you want to remove the selected header?' with 'Yes', 'No', and 'Cancel' buttons.

Name	Value	Entry Type
X-Powered-By	ASP.NET	Local

2.4 X-Content-Type-Options

Information:

Die X-Content-Type-Options verhindern, dass ein Browser versucht, den Inhaltstyp per MIME-Sniff zu ermitteln. Der Header zwingt ihn, sich an den deklarierten Content-Type zu halten und schützt davor, dass nicht ausführbare Inhalte in ausführbare umgewandelt werden. Mit dem Wert nosniff wird dieser Schutzmechanismus für Skripte und Stile aktiviert.

Lösung:

Name:	<input type="text" value="X-Content-Type-Options"/>
Value:	<input type="text" value="nosniff"/>



2.5 X-Frame-Options

Information:

Der HTTP-Antwort-Header X-Frame-Options kann verwendet werden, um anzugeben, ob ein Browser eine Seite in einem <frame>, <iframe>, <embed> oder <object> darstellen darf. Websites können dies nutzen, um Clickjacking-Angriffe zu vermeiden, indem sie sicherstellen, dass ihre Inhalte nicht in andere Websites eingebettet werden.

Clickjacking liegt vor, wenn ein Angreifer mehrere Ebenen verwendet, um einen Benutzer dazu zu verleiten, auf eine Schaltfläche oder einen Link zu klicken, die sich auf einer anderen Seite befinden, obwohl der Benutzer eigentlich auf die Seite der obersten Ebene klicken wollte. Der Angreifer "entführt" also Klicks, die für seine Seite bestimmt sind, und leitet sie auf eine andere Seite weiter, die höchstwahrscheinlich zu einer anderen Anwendung, Domäne oder beidem gehört.

Lösung:

Name:	<input type="text" value="X-FRAME-OPTIONS"/>
Value:	<input type="text" value="SAMEORIGIN"/>

X-FRAME-OPTIONS : DENY

Die Seite kann nicht in einen Frame gesetzt werden, egal wer es ist (einschließlich der Website, die selbst Frames verwendet). Wenn Sie auf Ihrer eigenen Website keine Frames verwenden, ist dies ein guter Aufhänger.

X-FRAME-OPTIONS : SAMEORIGIN

Die Seite kann geframt werden, solange die Domain, die sie framt, dieselbe ist. Das ist praktisch, wenn Sie selbst Frames verwenden.

X-FRAME-OPTIONS : ALLOW-FROM <https://myotherdomain.com>

Die Seite kann von den angegebenen Domänen geframt werden. Es ist optimal, wenn Sie zwei Websites haben. Auf diese Weise kann die eine Website von der Anderen geframt werden.

Hinweis: Der letzte Wert des X-Frame-Options Header "ALLOW-FROM uri" wird von modernen Browsern nicht mehr unterstützt.

2.6 X-XSS-Protection

Information:

1; mode=block aktiviert die XSS-Filterung. Anstatt die Seite zu säubern, verhindert der Browser das Rendern der Seite, wenn ein Angriff erkannt wird. Der Browser stoppt das Laden von Seiten, wenn er reflektierte Cross-Site-Scripting-Angriffe (XSS) erkennt.

XSS = Cross-site scripting

Cross-Site-Scripting (XSS) ist eine Sicherheitslücke, die es einem Angreifer ermöglicht, einen bösartigen und clientseitigen Code in eine Website einzuschleusen. Dieser Code wird von den Opfern ausgeführt und ermöglicht es den Angreifern, die Zugangskontrollen zu umgehen und sich als Benutzer auszugeben.

Lösung:

Name:	<input type="text" value="X-XSS-Protection"/>
Value:	<input type="text" value="1; mode=block"/>

2.7 Referrer-Policy

Information:

Der Referrer-Header enthält die Adresse einer Anfrage und hat viele relativ harmlose Verwendungszwecke, wie z. B. Analyse, Protokollierung oder optimierte Zwischenspeicherung. Es gibt jedoch auch problematischere Verwendungszwecke wie die Nachverfolgung, den Diebstahl von Informationen oder auch die unbeabsichtigte Weitergabe sensibler Informationen.

Abschließend fassen wir zusammen, dass es einer Website mithilfe der Referrer-Policy ermöglicht wird, die Anzahl an Informationen zu kontrollieren, welche bei der Navigation von einem Dokument durch den Browser einbezogen wird. Mit der Richtlinie no-referrer wird der Referrer-Header ganz weggelassen.

Lösung:

Name:	<input type="text" value="Referrer-Policy"/>
Value:	<input type="text" value="no-referrer"/>



2.8 Update der TLS und SSL Version

Information:

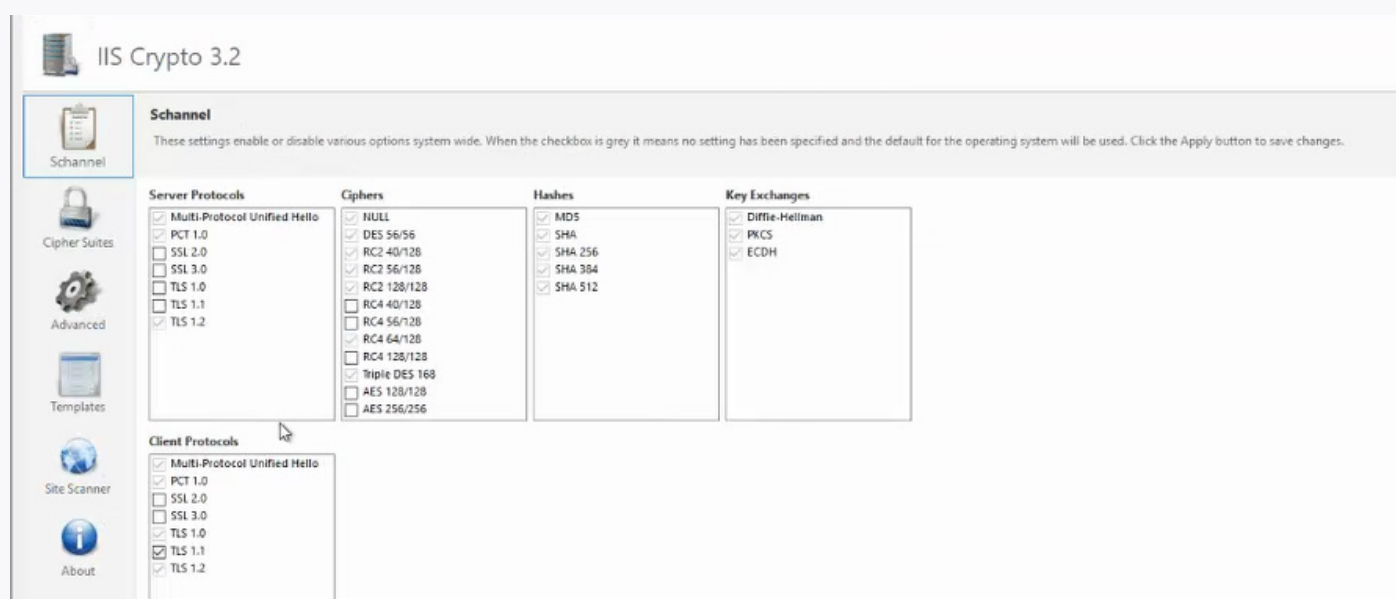
Die Transport Layer Security (TLS) und Secure Sockets Layer (SSL) sind Protokolle, die eine sichere Kommunikation bereitstellen. Active Directory-Verbunddienste (AD FS) verwendet diese Protokolle für die Kommunikation. Heute sind mehrere Versionen dieser Protokolle vorhanden.

Schannel ist ein Security Support Provider (SSP), der die standardmäßigen Internetauthentifizierungsprotokolle SSL, TLS und DTLS implementiert.

Microsoft: SSL/TLS-Protokolle und Cipher-Suites

Das Problem ist, dass mittlerweile einige Versionen als unsicher eingestuft werden und daher ein Upgrade benötigen. Unter der Website <https://www.ssllabs.com/ssltest/> kann man seinen Host testen und erhält daraufhin ein Rating mit Informationen darüber, was aktuell als sicher eingestuft wird. Dazu zählen unter anderem Zertifikate, ihr Ablaufdatum, der Aussteller und ob sie als „trusted“, also vertrauenswürdig, eingestuft werden können.

Unter <https://www.nartac.com/Products/IISCrypto/Download> erhält man die Software IIS Crypto. Mithilfe von IIS Crypto können Anpassungen sekundenschnell vorgenommen und bestimmte Versionen von Protokollen und Cipher Suites deaktiviert werden:



IIS Crypto 3.2

Cipher Suites
Enable, disable or reorder various cipher suites that are negotiated for the TLS handshake

Schannel

Cipher Suites

Advanced

Templates

Site Scanner

About

<input checked="" type="checkbox"/>	TLS_AES_256_GCM_SHA384	
<input checked="" type="checkbox"/>	TLS_AES_128_GCM_SHA256	
<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	
<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	
<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	
<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	
<input checked="" type="checkbox"/>	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	
<input checked="" type="checkbox"/>	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	
<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	
<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	
<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384	
<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	
<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	
<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	
<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	
<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	
<input type="checkbox"/>	TLS_RSA_WITH_AES_256_GCM_SHA384	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_AES_128_GCM_SHA256	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_AES_256_CBC_SHA256	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_AES_128_CBC_SHA256	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_AES_256_CBC_SHA	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_AES_128_CBC_SHA	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_3DES_EDE_CBC_SHA	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_NULL_SHA256	
<input checked="" type="checkbox"/>	TLS_RSA_WITH_NULL_SHA	
<input checked="" type="checkbox"/>	TLS_PSK_WITH_AES_256_GCM_SHA384	
<input checked="" type="checkbox"/>	TLS_PSK_WITH_AES_128_GCM_SHA256	
<input checked="" type="checkbox"/>	TLS_PSK_WITH_AES_256_CBC_SHA384	
<input checked="" type="checkbox"/>	TLS_PSK_WITH_AES_128_CBC_SHA256	
<input checked="" type="checkbox"/>	TLS_PSK_WITH_NULL_SHA384	
<input checked="" type="checkbox"/>	TLS_PSK_WITH_NULL_SHA256	

Aktuell empfohlene Einstellungen:

Protocols: TLS 1.3 und TLS 1.2

Best Practices Cipher Suite Order für Windows Server 2016 und höher:

```
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
```

2.9. Cache-Control

Cache-Control zwingt Caches den Request dem Ursprungs-Server zuzustellen, um die Gültigkeit zu validieren, bevor eine gecachte Kopie freigegeben wird.

Überprüfen Sie, dass die Antwort keine sensiblen, persönlichen oder benutzerspezifischen Informationen enthält. Wenn dies der Fall ist, sollten Sie die Verwendung der folgenden HTTP-Antwort-Header in Erwägung ziehen. Auf diese Weise wird die Speicherung und der Abruf des Inhalts aus dem Cache durch einen anderen Benutzer eingeschränkt oder verhindert:

Cache-Control: no-cache, no-store, must-revalidate, private

Expires: 0

Diese Konfiguration weist sowohl HTTP 1.0- als auch HTTP 1.1-konforme Cache-Server an, die Antwort nicht zu speichern und die Antwort (ohne Validierung) nicht aus dem Cache abzurufen, wenn eine ähnliche Anfrage eingeht.

2.10 Insecure HTTP Method – TRACE and TRACK

Die unsichere HTTP-Methode [TRACE] ist für diese Ressource aktiviert und kann ausgenutzt werden. Die Methoden TRACK und TRACE können von einem Angreifer verwendet werden, um Zugriff auf den Autorisierungs-Token/Sitzungscookie eines Anwendungsbenutzers zu erhalten, selbst wenn der Sitzungscookie durch das HttpOnly-Flag geschützt ist. Damit der Angriff erfolgreich ist, muss der Anwendungsbenutzer in der Regel einen älteren Webbrowser oder einen Webbrowser mit einer SOP-Umgehungsschwachstelle (Same Origin Policy) verwenden.

1. Öffnen Sie den IIS-Manager.
2. Wählen Sie die Website aus.
3. Doppelklicken Sie auf "Request Filtering" (Wenn Sie das Symbol für Request Filtering nicht sehen, installieren Sie es)
4. Gehen Sie zu "HTTP-Verbs".
5. Klicken Sie auf "Verb verweigern". Geben Sie "TRACE" ein. Klicken Sie auf "OK".
6. Klicken Sie im Menü "Aktionen" auf "Verb verweigern". Geben Sie "TRACK" ein. Klicken Sie auf "OK".
7. Klicken Sie auf "Verb verweigern". Geben Sie "OPTIONS" ein. Klicken Sie auf "OK".

The image shows four sequential screenshots from the IIS Manager interface, labeled 1 through 5, illustrating the steps to disable the TRACE and TRACK HTTP verbs.

- 1.** Shows the 'Request Filtering' icon in the left-hand navigation pane.
- 2.** Shows the 'HTTP Verbs' link selected in the left-hand navigation pane.
- 3.** Shows the 'Actions' pane with 'Deny Verb...' selected.
- 4.** Shows the 'Deny Verb' dialog box with 'TRACE' entered in the 'Verb' field and the 'OK' button highlighted.
- 5.** Shows the 'Request Filtering' configuration page with a table of verbs and their allowed status.

Verb	Allowed
TRACE	False
TRACK	False
OPTIONS	False



2.11 Content Security Policy

Information:

Hinweis: Der Content-Security-Policy-Header (CSP) ist die verbesserte Version des X-XSS-Protection-Headers und bietet eine zusätzliche Sicherheitsebene. Der Aufwand den CSP Header zu implementieren, ist weitaus höher als bei anderen Headern und kann mit Ihren Websiteplugins in Konflikt geraten.

Daher: Wenden Sie den CSP nur unter mehrmaligen Tests in Simulationsumgebungen und auf eigene Gefahr an.

Ziele von CSP:

- Kontrolle über Ressourcen, die angefordert, eingebettet und ausgeführt werden können
- Sandbox iframes und jede Ressource, die entweder gerahmt werden kann oder nicht
- Informationen darüber, wann eine Anwendung ausgenutzt wird oder sich anders verhält

CSP stellt eine Reihe von Standard-HTTP-Headern zur Verfügung, die es Website-Besitzern ermöglichen, zugelassene Inhaltsquellen zu deklarieren, die von Browsern auf diese Seite geladen werden dürfen. Zu den abgedeckten Typen gehören JavaScript, CSS, HTML-Frames, Schriftarten, Bilder und einbettbare Objekte wie Java-Applets, ActiveX, Audio- und Videodateien.

Der CSP-HTTP-Header ist ein tiefgreifender Verteidigungsmechanismus, mit dem Entwickler ihre Webanwendungen auf verschiedene Weise absichern können. Er reduziert den Schaden, den eine böswillige Injektion verursachen könnte, indem er zulässt, was für eine bestimmte Seite geladen werden kann, um sich gegen Cross-Site-Scripting-Angriffe zu schützen. Von einigen Ausnahmen abgesehen, umfassen Richtlinien meist die Angabe von Serverherkunft und Skript-Endpunkten.

Hauptmuster von XSS-Attacken: Einschleusen von Inline-Skripten in HTML-Code, die dann bei jeder Anfrage ausgeführt werden.

Lösung:

'self' = Verweist auf den Ursprung, von dem aus das geschützte Dokument bereitgestellt wird, einschließlich desselben URL-Schemas und derselben Portnummer. Sie müssen die einfachen Anführungszeichen einschließen. Einige Browser schließen Blob und Filesystem ausdrücklich von den Quellrichtlinien aus. Seiten, die diese Inhaltstypen zulassen müssen, können sie mit dem Attribut Data angeben.

Es gibt Direktiven, die JavaScript, Bilder, Objekte und Style-Sheets kontrollieren. Jede Direktive wird durch einen Quelltextausdruck definiert. Das Attribut "nonce" hat sich in der Praxis nicht durchsetzen

können. JavaScript-Bibliotheken von Drittanbieter-Seiten konnten nicht dargestellt werden. Die Direktive "strict-dynamic" versucht, dies zu lösen. "report-uri" ist jetzt "report-to".

Die gültigen Werte sind:

none

self

data:

Directives:

default-src = Standard für alle directives.

default-src 'self' = Alles zulassen, aber nur von derselben Herkunft.

script-src 'self' = Nur Skripte desselben Ursprungs zulassen

script-src 'nonce' = Zusätzliche Flexibilität bei JavaScript Darstellungen, script-src hat in der Vergangenheit nicht ausgereicht.

frame-ancestors = Hat bei modernen Browsern mittlerweile den X-Frame-Options Header mit dem Value "Allow-From" abgeöst. Beispiel: frame-ancestors 'self' individueller domainname;

script-src 'self' 'unsafe-inline' = Ermöglicht die Verwendung von Inline-Ressourcen, wie z. B. Inline-`<script>`-Elemente, javascript: URLs, Inline-Event-Handler und Inline-`<style>`-Elemente. Die einfachen Anführungszeichen sind erforderlich. Hinweis: Abgesehen von einem sehr speziellen Fall sollten Sie die Verwendung des unsafe-inline-Schlüsselworts in Ihrer CSP-Richtlinie vermeiden. Wie Sie sich denken können, ist die Verwendung von unsafe-inline im Allgemeinen unsicher. Das unsafe-inline-Schlüsselwort hebt die meisten der Sicherheitsvorteile auf, die die Content-Security-Policy bietet.

base-uri = schränkt die URLs ein, die im `<base>`-Element eines Dokuments verwendet werden können. Fehlt dieser Wert, so ist jeder URI zulässig.

connect-src = schränkt die URLs ein, die über Skriptschnittstellen geladen werden können

img-src = 'src' gibt den Pfad zum Bild an

font-src = gibt gültige Quellen für Schriftarten an, die mit `@font-face` geladen werden. Die CSS-At-Regel `@font-face` legt eine benutzerdefinierte Schriftart fest, mit der Text angezeigt werden soll. Die Schriftart kann entweder von einem entfernten Server oder von einer lokal auf dem Computer des Benutzers installierten Schriftart geladen werden.

child-src = definiert die gültigen Quellen für Web Worker und verschachtelte Browsing-Kontexte, die mit Elementen wie `<frame>` und `<iframe>` geladen werden.

media-src = spezifiziert gültige Quellen für das Laden von Medien unter Verwendung der `<audio>` und `<video>` Elemente.

style-src = legt gültige Quellen für Stylesheets fest.

object-src = spezifiziert gültige Quellen für die Elemente `<object>`, `<embed>` und `<applet>`.

plugin-types = schränkt die Menge der Plugins ein, die in ein Dokument eingebettet werden können, indem die Arten von Ressourcen, die geladen werden können, begrenzt werden.

report-uri = meldet Versuche, die gegen die Sicherheitsrichtlinie für Inhalte verstoßen.

Name:	<input type="text" value="Content-Security-Policy"/>
Value:	<input type="text" value="default-src 'self'; script-src 'unsafe-inline' 'self'; style-src 'unsafe"/>



```
default-src 'self'; script-src 'unsafe-inline' 'self'; style-src 'unsafe-inline' 'self'
```

Die Nutzung von Inline Funktionen für CSS oder JavaScript sollte grundsätzlich unterlassen werden
'unsafe-inline' innerhalb der script-src gilt als trivialer Fehler, ebenso 'self'

```
default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';
```

Der Einsatz von Bildern, Styles (CSS) und JavaScript wird von der eigenen Domäne erlaubt. Nutzt man JavaScript, bedeutet das, dass man nur eingebundene Script-Dateien ausführen darf. Inline-Anweisungen, die in einem Script-Tag stehen, sind nicht erlaubt. Ein Problem entsteht hier dann, wenn eine Webanwendung so entwickelt wurde, dass direkt in HTML-Code mit dem Element script gearbeitet wird.

Damit nicht die ganze Website umgeschrieben werden muss, wird in der Regel die Direktive um unsafe-inline erweitert. Die Webanwendung arbeitet wie gewohnt, ist jedoch nicht anfällig auf Reflected- oder Stored-XSS-Angriffe. Das ergibt aber weniger Sinn, weil die CSP somit weniger schützend wirkt.

Wenn externe JavaScript Bibliotheken eingebunden werden, müssen diese Domänen über Whitelisting in der Direktive script-src gepflegt werden.

script-src zu verwenden wird nicht empfohlen. Grund: Ausführung von Inline-JavaScript wird weiterhin zugelassen. Daher bietet die Policy so keinen reellen Schutz vor XSS-Angriffen.

„nonce“ ermöglicht eingebettete Script-Elemente zu nutzen. Dafür wird ein zufälliger Wert in die Direktive geschrieben. Dies geschieht durch die Webanwendung selbst. Jedes Script-Element, welches diesen Wert im Attribut nonce einsetzt, wird danach zugelassen. So können die legitimen Script-Elemente ausgeführt werden, jedoch werden XSS-Angriffe gestoppt.

CSP nicht in Kombination mit CDNs hosting Angular JS verwenden: Beim Einsatz von Bibliotheken, unter anderem JavaScript-CDNs wie jQuery, React.js oder AngularJS, ergibt sich mit der Verwendung von nonce ein neues Problem, das nachfolgend aufgezeigt wird. Das Attribut nonce wird entsprechend definiert:

```
Content-Security-Policy: script-src 'nonce-i7bGtfs'
```

Da der korrekte Wert für nonce gesetzt ist, wird das Einbinden der Bibliothek erlaubt. Wenn nun **aber in der Bibliothek selbst auch noch weitere Bibliotheken als Abhängigkeit nachgeladen werden oder ein Script-Element verwendet wird, führt dies zu einem Fehler**, da dort der Wert für nonce nicht gesetzt ist.

Wenn man default-src oder script-src + strict-dynamic und nonce verwendet, werden Whitelists und die Verwendung von unsafe-inline verworfen.

script-src ist am wichtigsten

Damit png's geladen werden:

```
default-src 'self'; script-src 'self'
```

Hauptproblem mit Firefox: Implementiert keine plugin-types, unterstützt child-src nicht.



2.12 Permissions Policy Header

Information:

Der Permissions Policy Header ist eine zusätzliche Sicherheitsebene, die hilft, den unbefugten Zugriff oder die Nutzung von Browser-/Client-Funktionen durch Webressourcen zu beschränken. Diese Richtlinie gewährleistet den Schutz der Privatsphäre des Benutzers, indem sie die Funktionen des Browsers einschränkt oder festlegt, welche Funktionen von den Webressourcen verwendet werden können. Die Berechtigungsrichtlinie bietet eine Reihe von Standard-HTTP-Headern, die es den Website-Besitzern ermöglichen, Browserfunktionen einzuschränken, die von der Seite verwendet werden können, z. B. Kamera, Mikrofon, Standort, Vollbild usw.

Lösung:

Ebenfalls im IIS HTTP Response Headers anzulegen.

Google hat beispielsweise folgende Parameter:

permissions-policy	ch-ua-arch=*, ch-ua-bitness=*, ch-ua-full-version=*, ch-ua-full-version-list=*, ch-ua-model=*, ch-ua-wow64=*, ch-ua-platform=*, ch-ua-platform-version=*
--------------------	--

Es sind zwei große Klassen von Hinweisen definiert: Hohe Entropie und niedrige Entropie. High-Entropy-Hinweise enthalten eine größere Menge an potentiell identifizierenden Informationen über den Benutzer und sind nicht für die automatische Verteilung an herkunftsübergreifende Ressourcen gedacht. Hinweise mit niedriger Entropie enthalten Informationen über den Browser, von denen erwartet wird, dass sie über große Gruppen von Nutzern hinweg konsistent sind (z. B. Browsername oder Hauptversion), und sind für die Identifizierung einzelner Nutzer weitaus weniger nützlich.

Hinweise mit hoher Entropie werden durch Merkmale mit einer Standard-Zulassungsliste von selbst unterstützt. Hinweise mit geringer Entropie werden von Merkmalen mit einer Standard-Erlaubnisliste von * unterstützt..

Beispiel für hochentropische Hinweise:

DPR (ch-dpr)

UA-Arch (ch-ua-arch)

Beispiel für Hinweise mit geringer Entropie:

Daten speichern (ch-save-data)

UA (ch-ua)

2.13 Internal IP address disclosed

Information:

Diese Offenlegung kann Informationen über das IP-Adressierungsschema des internen Netzwerks eines Unternehmens/einer Organisation preisgeben. Ein Angreifer kann die IP-Adressen verwenden, um weitere Angriffe auf bestimmte Benutzer durchzuführen. Es gibt viele Möglichkeiten, wie ein Angreifer die private IP-Adresse der Benutzer ermitteln kann.

Lösung:

Microsoft IIS Interne IP-Adressoffenlegung Sicherheitslücke. Führen Sie die folgenden Schritte durch, um die Offenlegung interner IP-Adressen zu verhindern:

- Öffnen Sie eine Eingabeaufforderung und ändern Sie das aktuelle Verzeichnis in `c:\inetpub\adminscripts` oder in das Verzeichnis, in dem die `adminscripts` zu finden sind.
- Führen Sie folgende Befehle aus:

```
adsutil set w3svc/UseHostName True
net stop iisadmin /y
net start w3svc
```

Dadurch wird der IIS-Server veranlasst, den Hostnamen des Rechners anstelle der IP-Adresse zu verwenden.

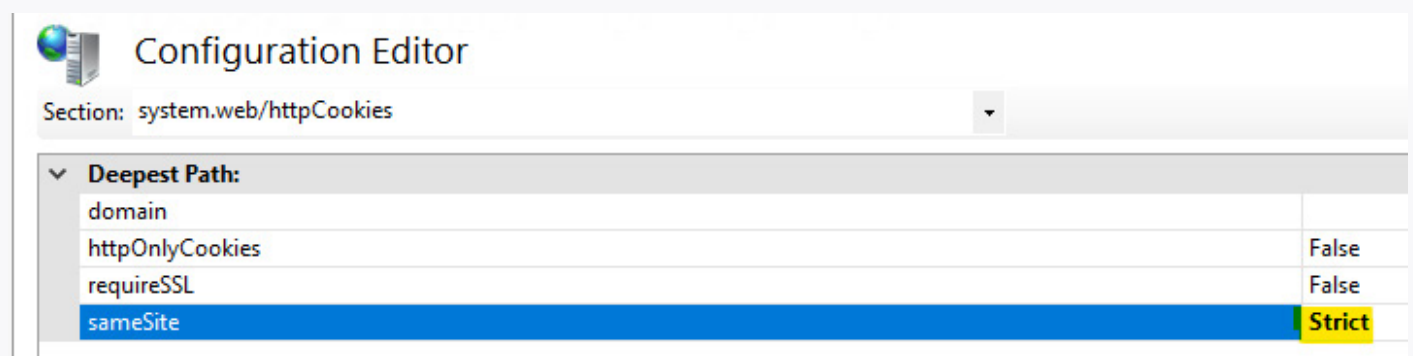
2.14 Cookie SameSite Attribute

Information:

Es wurde ein Cookie ohne das SameSite-Attribut gesetzt, was bedeutet, dass dieser Cookie als Ergebnis einer "Cross-Site"-Anfrage gesendet werden kann. Das SameSite-Attribut ist eine wirksame Gegenmaßnahme gegen Cross-Site Request Forgery, Cross-Site Script Inclusion und Timing-Angriffe.

Lösung:

Setzen Sie im "Configurations-Editor" unter `system.web/httpCookies` den `sameSite`-Wert auf "Strict" (idealerweise) oder "Lax".



The screenshot shows the Configuration Editor window. The title bar reads "Configuration Editor". Below the title bar, the "Section:" dropdown is set to "system.web/httpCookies". The main content area shows a tree view with "Deepest Path:" expanded. Underneath, there is a table with the following attributes and values:

domain	
httpOnlyCookies	False
requireSSL	False
sameSite	Strict

Diese Einstellungen sollten so schon als Default gesetzt sein, demnach besteht hierbei für Sie kein Handlungsbedarf, außer Sie hatten bereits schon selbst Änderungen vorgenommen.

system.web/anonymousIdentification cookieRequireSSL Wert auf "false" setzen.

system.web/sessionState: cookieSameSite auf "Lax",

system.web/roleManager: cookieRequireSSL auf "false".

```
XML Kopieren  
  
<configuration>  
  <system.web>  
    <anonymousIdentification cookieRequireSSL="false" /> <!-- No config attribute for S...>  
    <authentication>  
      <forms cookieSameSite="Lax" requireSSL="false" />  
    </authentication>  
    <sessionState cookieSameSite="Lax" /> <!-- No config attribute for Secure -->  
    <roleManager cookieRequireSSL="false" /> <!-- No config attribute for SameSite -->  
  </system.web>  
</configuration>
```

2.15 Anti-CSRF Tokens

<cross-site
request forgery>

Information:

Bei einer Cross-Site Request Forgery handelt es sich um einen Angriff, bei dem ein Opfer ohne sein Wissen oder seine Absicht gezwungen wird, eine HTTP-Anfrage an ein Ziel zu senden, um eine Aktion durchzuführen. Die zugrundeliegende Ursache ist die Anwendungsfunktionalität, die vorhersehbare URL-/Formular-Aktionen auf wiederholbare Weise verwendet. Das Wesen des Angriffs besteht darin, dass CSRF das Vertrauen ausnutzt, das eine Website bei einem Benutzer genießt. Im Gegensatz dazu wird beim Cross-Site-Scripting (XSS) das Vertrauen des Benutzers in eine Website ausgenutzt. Wie XSS sind auch CSRF-Angriffe nicht notwendigerweise seitenübergreifend, aber sie können es sein. Cross-Site Request Forgery ist auch als CSRF, XSRF, Ein-Klick-Angriff, Session Riding, Confused Deputy und Sea Surf bekannt.

- Muss im HTML-Übermittlungsformular enthalten sein.
- Drittgrößter Web-Angriff, besonders anfällig: Banken.

Voraussetzung ist, dass Webbrowser dem Code vertrauen. Wenn sie ihn erhalten, werden sie ihn ohne weiteres ausführen.



Beispiel:

```
<form role="search" class="searchform fusion-search-form fusion-search-form-clean" method="get" action="https://gbs.com/">
```

Besonders anfällig:

Get: Copy&Paste, es bleibt alles beim Alten. Statisch.

Post: Schreiben von Kontexten, die einmal geschehen, wenn man auf Aktualisieren drückt und zu einer Seite zurückgeht, wird dasselbe nicht zweimal ausgeführt. Dynamisch.

Szenarien:

Original Bank -> Formular -> per "Post" wird eine Anfrage gesendet

Gefälschte Bank-Seite -> dupliziertes Formular

Gefälschter Blog, hinterlegtes Formular mit Kontonummer und Transaktionsbetrag, wenn jemand kommentiert, wird die Anfrage an eine bösartige Seite weitergeleitet.

Das funktioniert, weil das Opfer gerade in einem anderen Tab bei seiner Online-Bank eingeloggt ist und dieser autorisiert ist. Die Bank erhält die Anfrage und sieht sich die seltsamen Blog-Parameter an, kümmert sich aber nicht darum, weil das Formular mit den Bankdaten versteckt ist.

Früher wurde eine Seite komplett neu geladen, nachdem man einen Kommentar abgegeben und auf "Posten" geklickt hatte. Heutzutage geschieht das im Hintergrund – der User merkt es nicht und genau das kann zur Gefahr werden.

Lösung:

Verwenden Sie eine geprüfte Bibliothek oder ein Framework, das diese Schwachstelle nicht zulässt oder Konstrukte bereitstellt, mit denen diese Schwachstelle leichter zu vermeiden ist, z. B. Anti-CSRF-Pakete wie OWASP CSRFGuard.

Stellen Sie sicher, dass Ihre Anwendung frei von Cross-Site-Scripting-Problemen ist, da die meisten CSRF-Schutzmaßnahmen mit einem vom Angreifer kontrollierten Skript umgangen werden können.

Generieren Sie eine eindeutige Nonce für jedes Formular, fügen Sie die Nonce in das Formular ein und verifizieren Sie die Nonce beim Empfang des Formulars. Achten Sie darauf, dass die Nonce nicht vorhersehbar ist. Verwenden Sie die keine GET-Methode für eine Anfrage, die eine Zustandsänderung auslöst.

Wenn Ihre Website einen einfachen Anti-CSRF-Token verwendet, setzt der Webserver diesen Token in den Sitzungscookie Ihres Webbrowsers, gleich nachdem Sie sich angemeldet haben. Alle Formularübermittlungen enthalten ein verstecktes Feld, das den Token enthält. Dadurch wird die CSRF-Schwachstelle vollständig beseitigt.

Siehe 2.7 Referrer Policy

Probleme für einige Benutzer: Der Referrer-Header wird nicht immer so gesendet, wie es sein sollte. Einige Adverb-Blocker oder ein Datenschutz-Tool blockieren ihn. Daher wird ein "Nonce" als ein einmaliger Schlüssel verwendet.

Nonce = ein Formular auf einer Website generiert einen eindeutigen Code (alles, was der Nutzer dann sieht, muss diesen Token haben). Dieser Token ist ebenfalls völlig unsichtbar und besteht aus zufälligen Zeichenfolgen.

Microsoft hat einen eingebauten CSRF-Schutz in das neue Web-Formular-Anwendungsprojekt integriert:

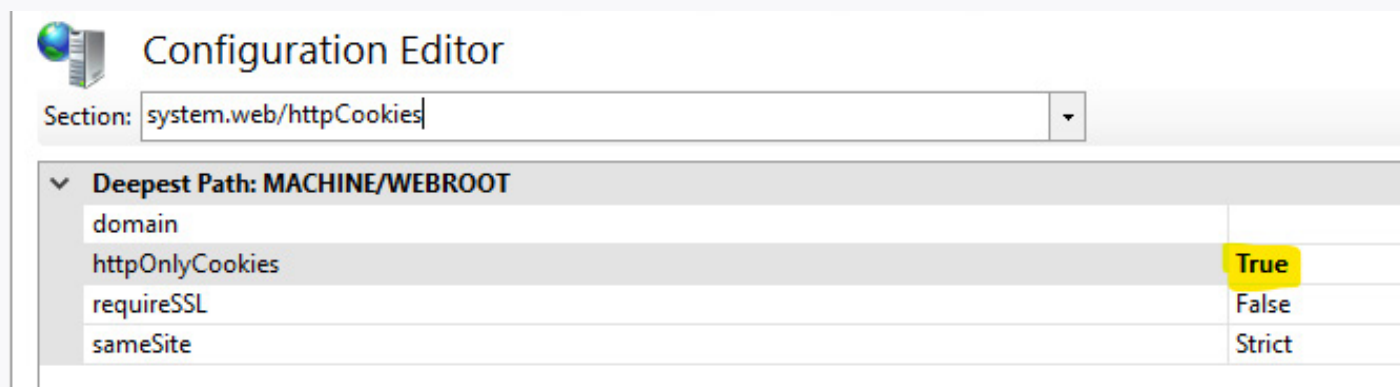
```
if ((context.Request.UrlReferrer == null || context.Request.Url.Host != context.Request.UrlReferrer.  
Host))  
{  
    context.Response.Redirect("~/error.aspx", false);  
}
```

Cookie Access beschränken: HttpOnly flag

Set-Cookie: AuthCookie=insertyourcookiesesshere=;/; HttpOnly

Ziehen Sie das SameSite-Cookie-Attribut für Sitzungscookies in Betracht, aber achten Sie darauf, KEIN Cookie speziell für eine Domäne zu setzen, da dies eine Sicherheitslücke darstellen würde, die dazu führt, dass alle Subdomänen dieser Domäne diesen Cookie gemeinsam nutzen. Dies ist insbesondere dann ein Problem, wenn eine Subdomain einen CNAME zu Domains hat, die nicht unter Ihrer Kontrolle stehen.

„HttpOnly“: Markieren Sie den Sitzungscookie immer als "HTTPOnly", damit es nicht mit JavaScript aufgerufen werden kann.



The screenshot shows the Configuration Editor interface. The 'Section' dropdown is set to 'system.web/httpCookies'. The 'Deepest Path' is 'MACHINE/WEBROOT'. A table lists the following settings:

Property	Value
domain	
httpOnlyCookies	True
requireSSL	False
sameSite	Strict

Secure Flag im IIS:

Besser ist es, URL Rewrite zu verwenden und die Datei web.config wie folgt zu ergänzen: <system.webServer>

```
<rewrite>
<outboundRules>
<rule name="Use only secure cookies" preCondition="Unsecured cookie">
<match serverVariable="RESPONSE_SET_COOKIE" pattern=".*" negate="false" />
<action type="Rewrite" value="{R:0}; secure" />
</rule>
<preConditions>
<preCondition name="Unsecured cookie">
<add input="{RESPONSE_SET_COOKIE}" pattern="." />
<add input="{RESPONSE_SET_COOKIE}" pattern=";" secure="true" negate="true" />
</preCondition>
</preConditions>
</outboundRules>
</rewrite>
...
</system.webServer>
```

Achtung: Bei der Verwendung moderner Frameworks ist ein auf Token basierender CSRF Schutz in der Regel bereits enthalten oder kann leicht zu Formularen hinzugefügt und von der entsprechenden Middleware validiert werden.

Der Same-Site-Cookie-Ansatz schränkt den Ursprung ein, von dem ein Cookie gesendet werden kann. Auf diese Weise nutzt Cross-Site Request Forgery die Möglichkeit aus, eine herkunftsübergreifende Anfrage (und damit seitengleiche Cookies) zu stellen. Durch die Einschränkung, dass Anfragen nur von dem Ursprung aus gesendet werden können, mit dem ein Cookie verknüpft ist, wird die Möglichkeit, externe Anfragen an eine Anwendung zu senden, effektiv verhindert.



3. Security-Header Analyse

Fragen Sie sich gerade, wie Ihre Website hinsichtlich Ihrer eigenen Security HTTP Header aufgestellt ist? Schnell und einfach können Sie unter den folgenden Links Ihre Leistung herausfinden und dann entsprechende Schritte einleiten, um Ihre Website sicherer zu machen.

Security Header:

<https://securityheaders.com>

<https://www.serpworx.com>

SSL/Server Test:

<https://www.ssllabs.com/ssltest/>

Sprechen Sie uns bei Fragen an. Wir helfen Ihnen gerne weiter die Sicherheit Ihres Unternehmens zu verbessern

Ihr GBS-Team

Autor:

Anya Raider

Cyber Security Consultant bei GBS

„Als Cyber Security Consultant helfe ich GBS und unseren Kunden bei der Auswahl der richtigen Sicherheitslösungen für ihre individuellen Anforderungen. Ich habe in verschiedenen Sektoren der IT gearbeitet und habe daher viele verschiedene Szenarien und Fälle in der Branche erlebt. Ich freue mich immer, wenn sich die Sicherheitslage eines Kunden verbessert und er dadurch besser gegen bösartige Akteure und fortschrittliche Bedrohungen geschützt ist.“

Über GBS

GBS ist ein anerkannter Anbieter von E-Mail- und Collaboration-Sicherheitslösungen in Deutschland mit fast 30 Jahren Erfahrung in den Bereichen Datenschutz, Produktivität und Compliance. Das Unternehmen wird von namhaften Marktforschern in Deutschland und von seinen Partnern als führendes Unternehmen für Cybersicherheitslösungen anerkannt, mit Schwerpunkt in den Bereichen Data Loss Prevention und Collaboration Sicherheit.

GBS bietet umfangreiche Lösungen der nächsten Generation für E-Mail-Produktivität, Compliance sowie einen mehrstufigen Schutz bei der E-Mail-Kommunikation und Daten-austausch über verschiedene Collaboration-Plattformen gegen alle Arten von Sicherheits-bedrohungen. Die Lösungen für Microsoft 365, Exchange und HCL Domino sind einfach zu bedienen, flexibel und decken Schlüsselbereiche wie Malware-Schutz, Verschlüsselung, E-Mail-Produktivität, Datenverluste, Workflow und Compliance ab.

Die Lösungen von GBS schützen mehr als 2 Millionen Endbenutzer weltweit. Das Unter-nehmen hat langjährige Beziehungen zu über 2.000 Kunden aufgebaut.