Sociedad Andaluza para el Desarrollo de las Telecomunicaciones
CONSEJERÍA DE EMPLEO, EMPRESA Y COMERCIO

seguridad+ Y CONFIANZA DIGITAL | AndalucíaCERT CENTRO DE SEGURIDAD TIC

Informe de divulgación
OWASP Secure Headers Project

Tipo de documento:	*Informe*
Autor del documento:	*AndalucíaCERT*
Código del Documento:	*CERT-IF-10086-170317*
Edición:	*0*
Categoría	*Público*
Fecha de elaboración:	*08/06/2017*
N° de Páginas	*1 de 14*

1 TABLA DE CONTENIDOS

2 OBJETIVO

El objeto de este documento es dar a conocer a los usuarios el OWASP Secure Headers Project, proyecto que gira en torno a la seguridad en el protocolo HTTP. Se centra fundamentalmente en introducir una serie de cabeceras HTTP seguras en el lado servidor, produciendo así una mejora notable en la seguridad de las comunicaciones.

3 ALCANCE

Este documento va dirigido tanto al personal de la Junta de Andalucía, como al público en general. Es de carácter divulgativo e informativo, en el cual se pretende dar a conocer el OWASP Secure Headers Project, de manera que el usuario sea consciente de cómo es posible aumentar considerablemente la seguridad en las comunicaciones cliente-servidor HTTP aplicando una serie de directrices sencillas.

4 INTRODUCCIÓN.

HTTP es el protocolo de comunicación que rige cómo se transfiere la información en el marco de la World Wide Web. HTTP define la sintaxis y la semántica que utilizan los elementos de software dentro de la arquitectura web (clientes, servidores, proxies) para comunicarse. Es un protocolo orientado a transacciones, por lo que se basa en un modelo cliente-servidor en el que se sigue un esquema de petición-respuesta.

4.1 Las cabeceras HTTP

El cliente (se le conoce "agente de usuario") realiza una petición enviando un mensaje con cierto formato al servidor. El servidor (se le llama normalmente servidor web) le envía un mensaje de respuesta. Las **cabeceras** HTTP son los parámetros que se envían en una petición o respuesta HTTP al cliente o al servidor para proporcionar información esencial sobre la transacción en curso. Estas cabeceras proporcionan información mediante la sintaxis 'Cabecera: Valor' y son enviadas automáticamente por el navegador o el servidor Web.

Existe una amplia selección de cabeceras, entre las que destacan algunas como:

Accept: Content-Types (tipos de contenido) que se aceptan.
Accept: text/plain

Accept-Language: Idiomas que que se aceptan.
Accept-Language: en-US

Cookie*:* Una cookie enviada previamente por el servidor usando Set-Cookie.
Cookie: $Version=1; Skin=new;

Date: La fecha y la hora de la petición.
Date: Tue, 15 Nov 1994 08:12:31 GMT

Referer*:* Indica la dirección URL de donde proviene, en otras palabras, es la dirección web del botón Atrás.
Referer:https://www.owasp.org/index.php/

User-Agent*:* Contiene la información de la petición, como el navegador, el sistema operativo, etc.
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0

Las cabeceras le dan gran flexibilidad al protocolo permitiendo añadir nuevas funcionalidades sin tener que cambiar la base. Por eso según han ido sucediendo las versiones de HTTP se han ido permitiendo más cabeceras.

Hay que mencionar que dependiendo del tipo de mensaje en el que se integre una cabecera las podemos clasificar en cabeceras de petición, cabeceras de respuesta y cabeceras que pueden ir tanto en un petición como en una respuesta (generalmente orientadas a conexión). Este documento va a versar sobre cabeceras de respuesta (lado del servidor) orientadas a hacer segura la comunicación (OWASP Secure Headers Project).

4.2 Fundación OWASP

OWASP (Open Web Application Security Project, en español 'Proyecto abierto de seguridad de aplicaciones web') es un proyecto de código abierto enfocado a determinar y combatir las causas que hacen que el software web sea inseguro. La Fundación OWASP es un organismo sin ánimo de lucro que cuenta con un amplio grupo de colaboración formado por empresas, organizaciones educativas y particulares de todo mundo, constituyendo juntos una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se publican y pueden ser usadas gratuitamente por cualquiera.

La forma de trabajar de OWASP es la de intentar cubrir la mayor cantidad de ámbitos dentro de la seguridad informática. Para ello divide su actividad en torno a proyectos que conforman un vasto conjunto de herramientas, documentos, entornos de prueba y cualquier otro material que ayude a las organizaciones a producir código seguro.

De todos los proyectos OWASP (hay en torno a 93 actualmente), el más conocido es posiblemente el OWASP Top 10, que intenta ofrecer una lista de los 10 riesgos más importantes en la actualidad en el ámbito de las aplicaciones web. De esa manera se intenta educar a los desarrolladores, diseñadores, arquitectos, gerentes y organizaciones sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web. El Top 10 provee técnicas básicas sobre cómo protegerse en estas áreas de alto riesgo, ofreciendo también orientación sobre los pasos a seguir.

En este documento se va a explicar en qué consiste el **OWASP Secure Headers Project**, profundizando en las recomendaciones de seguridad que plantea y mostrando qué es y cómo se implementa cada una.

5 OWASP Secure Headers Project.

El OWASP Secure Headers Project se basa en la idea de que establecer cabeceras HTTP en el lado servidor es fácil y no requiere cambios en el código. Una vez establecidas, dichas cabeceras pueden restringir a los navegadores modernos de manera que no caigan en vulnerabilidades fácilmente prevenibles. El OWASP Secure Headers Project busca aumentar la conciencia entre aquellos que se dedican al ámbito web y fomentar el uso de la siguiente lista de cabeceras propuestas:

- HTTP Strict Transport Security (HSTS)
- Public Key Pinning Extension for HTTP (HPKP)
- X-Frame-Options
- X-XSS-Protection
- X-Content-Type-Options
- Content-Security-Policy
- X-Permitted-Cross-Domain-Policies
- Referrer-Policy

5.1 HTTP Strict Transport Security (HSTS)

HTTP Strict Transport Security (HSTS) es una política de seguridad web que permite proteger al sitio web frente a ataques de *Downgrade* y de secuestro de sesión (*Session Hijacking*). HSTS permite a los servidores web forzar que la interacción con los navegadores (o cualquier otro User-Agent) solo se produzca mediante conexiones HTTPS y nunca a través de HTTP, que es inseguro al utilizar comunicaciones en claro. HSTS consta de dos valores:

Valor	Descripción
max-age=SECONDS	El tiempo en segundos que el navegador debe recordar que a este sitio solo se puede acceder mediante HTTPS.
includeSubDomains	Parámetro opcional, si se habilita indica que la regla aplica también a todos los subdominios.

Un ejemplo de una cabecera HSTS puede ser:

Strict-Transport-Security: max-age=31536000 ; includeSubDomains

Si una aplicación web fuerza a los User-Agents a respetar la política HSTS, los User-Agents se comportarán de la siguiente manera:

- Inmediatamente cambian cualquier enlace inseguro en un enlace seguro.

- Si la seguridad de la comunicación no puede ser garantizada (por ejemplo, si el certificado TLS del servidor no es de confianza), se muestra un mensaje de error y no se permite al usuario el acceso a la aplicación web.

5.2 Public Key Pinning Extension for HTTP (HPKP)

HTTP Public Key Pinning (HPKP) es un mecanismo de seguridad que permite a los sitios web HTTPS resistir ataques de suplantación (*Man-in-the-Middle*). Estos ataques se producen a través de certificados comprometidos o certificados fraudulentos (en ocasiones los atacantes pueden comprometer alguna autoridad de certificación y emitir certificados falsos para un cierto sitio web).

HPKP se basa en la infraestructura de clave pública (X.509) con la que se crean los certificados en los sitios web. Si el servidor utiliza HPKP se lo comunicará al cliente a través de la cabecera de respuesta HTTP, y en ella incluirá una lista de hashes de clave pública que permiten desencriptar la cadena de certificados del sitio web. Se considerará válida la cadena de certificados si al menos uno de ellos puede descifrarse con una de las claves públicas recibidas por HPKP. Con esto se consigue demostrar la autoría de los certificados (ya que para que sean legítimos debieron ser firmados con la clave privada correspondiente a la pública). Hay que mencionar que el hash (o grupo de hashes) recibido por HPKP tiene una validez temporal determinada.

Los valores a la hora de usar HPKP son:

Valor	Descripción
pin-sha256="<sha256>"	La cadena entre comillas es la huella digital de la Información de Clave Pública del Sujeto (SPKI) codificada en base64. Es posible fijar varias claves públicas válidas a la vez. Algunos navegadores puede que permitan en el futuro otros métodos de hash distintos a SHA256.
max-age=SECONDS	El tiempo, en segundos, que el navegador debe recordar que solo puede acceder a este sitio usando una de las claves recibidas por HPKP.
includeSubDomains	Parámetro opcional, si se activa aplica la regla a todos los subdominios también.
report-uri="<URL>"	Parámetro opcional, si se activa los accesos fallidos se reportan a la URL especificada.

Un ejemplo de aplicación de cabecera HPKP puede ser:

Public-Key-Pins: pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM="; pin-sha256="E9CZ9INDbd+2eRQozYqqbQ2yXLVKB9+xcprMF+44U1g="; report-uri="http://example.com/pkp-report"; max-age=10000; includeSubDomains

El desarrollo de HPKP puede requerir de importante madurez logística, ya que podría darse el caso de que las claves públicas comunicadas por los hosts caduquen provocándose a sí mismos indisponibilidad en el servicio. Es decir, debe haber un control exhaustivo por parte de los hosts en torno a la validez temporal de las claves públicas que envían a los clientes.

5.3 X-Frame-Options

X-Frame-Options es una cabecera de respuesta que mejora la protección de las aplicaciones web frente a ataques de *Clickjacking.* Lo consigue a través de una política transmitida desde el servidor al navegador del cliente en la que se especifica que el navegador por defecto nunca debe mostrar el contenido transmitido en frames (marcos o cuadros) de otras páginas webs.

Los posibles valores que puede tomar X-Frame-Options son:

Valor	Descripción
deny	No permite representación dentro de un frame (marco)
sameorigin	No permite la representación dentro de un frame (marco) si el origen no coincide.
allow-from: DOMAIN	Permite la representación si el marco donde se incluye pertenece al mismo dominio.

Un ejemplo sería:

X-Frame-Options: deny

5.4 X-XSS-Protection

Esta cabecera activa el filtro frente a *Cross-Site Scripting* en el navegador. Hay que mencionar que actualmente solo Google Chrome, Safari y Microsoft's Internet Explorer la soportan, ya que son los únicos navegadores que cuentan con filtro para protección frente a *Cross-Site Scripting*. Los valores que podemos encontrar para esta cabecera son:

Valor	Descripción
0	Filtro deshabilitado.
1	Filtro habilitado. Si un ataque de cross-site scripting es detectado, el nave-

	gador desinfectará la página para parar el ataque.
1; mode=block	Filtro habilitado. Cuando se detecte un ataque de cross-site, más que desinfectar la página, el navegador bloqueará la reproducción de la página.
1; report=http://[YOURDOMAIN]/your_report_URI	Filtro habilitado. El navegador desinfectará la página e informará del ataque en la URI especificada.

Un ejemplo de uso sería:

X-XSS-Protection: 1; mode=block

Los filtros XSS de los navegadores que los traen implementados, siguen una forma de proceder basada en la idea de lista negra, es decir, intentan identificar patrones de caracteres o etiquetas considerados peligrosos dentro de los parámetros de petición que reciben las páginas web.

5.5 X-Content-Type-Options

El uso de esta cabecera evitará que el navegador represente tipos de contenidos distintos a lo establecido en el Content-Type de las cabeceras HTTP. Protege frente a ataques de *Cross-Site Scripting* que se aprovechan de la práctica llevada a cabo por algunos navegadores conocida como *Content Sniffing*. (o *MIME Sniffing*). Cuando los metadatos no dejen claro el tipo de contenido a representar, el navegador inspeccionará el flujo de datos para intentar deducir a qué tipo de formato de fichero pertenecen dichos datos. Esta práctica puede derivar en una vulnerabilidad de seguridad, ya que el atacante puede lograr engañar al navegador para que interprete un tipo de datos como otro distinto (por ejemplo, embebiendo un código javascript en una supuesta imagen), confundiendo al algoritmo de *MIME Sniffing*. La finalidad más habitual de llevar a cabo la confusión del algoritmo de *MIME Sniffing* es la de provocar ataques de *Cross-Site Scripting*.

En el caso de que se active X-Content-Type-Options, el único valor que puede tomar es:

Valor	Descripción
nosniff	Evita que el navegador lleve a cabo MIME Sniffing

Un ejemplo de funcionamiento sería:

X-Content-Type-Options: nosniff

Si se activa X-Content-Type-Options el navegador representará el contenido exactamente con el tipo que haya establecido el servidor, suprimiendo así el *Content Sniffing*.

5.6 Content Security Policy (CSP)

Este mecanismo se basa en la elaboración de políticas de seguridad que limitan el contenido a mostrar por el navegador. La cabecera CSP permite definir fuentes de confianza de dónde el navegador puede cargar contenido. Requiere un ajuste cuidadoso y una definición precisa de la política, ya que si está activado tiene un impacto alto en la forma en que el navegador muestra la página web. CSP establece una serie de reglas o políticas en cuanto al contenido que se muestra (o no) en la página web, previniendo así un amplio rango de ataques, incluyendo aquellos de *Cross-Site Scripting* y otros tipos de inyecciones Cross-Site. CSP no se considera una herramienta de primera línea de defensa, sino de defensa en profundidad, ya que permite establecer una amplia gama de restricciones al contenido mostrado.

Los posibles valores a considerar son:

Directiva	Descripción
base-uri	Establece la uri base para las uris relativas..
default-src	Establece la política de carga para todos los tipos de recursos que no tengan definida una directiva asociada.
script-src	Establece qué scripts puede ejecutar el recurso protegido.
object-src	Establece de dónde puede el recurso protegido cargar plugins.
style-src	Establece qué estilos (CSS) aplica el usuario al recurso protegido.
img-src	Establece desde dónde el recurso protegido puede cargar imágenes.
media-src	Establece de dónde puede cargar audio y vídeo el recurso protegido.
frame-src	Obsoleto y sustituido por child-src. Establece de dónde puede integrar frames el recurso protegido.
child-src	Establece de dónde puede integrar frames el recurso protegido.
frame-ancestors	Establece dónde puede integrarse el recurso protegido como un frame.
font-src	Establece desde dónde el recurso protegido puede cargar fuentes.
connect-src	Establece las URIs que el recurso protegido puede cargar usando interfaces de script.
manifest-src	Establece de dónde puede cargar el recurso protegido el manifiesto.
form-action	Establece las URIs que pueden ser usadas como acción de los elementos de formulario HTML.
sandbox	Especifica una política de sandbox HTML que el User-Agent aplica al recurso protegido.
script-nonce	Establece la ejecución de scripts en función de si se encuentra el atributo nonce

	dentro de los elementos del script.
plugin-types	Establece el grupo de plugins que pueden ser invocados por el recurso protegido limitando los tipos de recursos que pueden ser integrados.
reflected-xss	Instruye al User-Agent sobre cuándo activar o desactivar heurísticas para filtrar o bloquear ataques de Cross-Site Scripting reflejado. Equilavente a la acción de la cabecera X-XSS Protection cuando se encuentra habilitada.
block-all-mixed-content	Impide que el User-Agent cargue contenido mixto (contenido seguro a través de HTTPS e inseguro a través de HTTP).
upgrade-insecure-reques-ts	Indica al User-Agent para que descargue recursos inseguros usando HTTPS.
referrer	Establece la información que el User-Agent debe enviar en el Referer.
report-uri	Establece la URI a la cuál el User-Agent envía los informes sobre violaciones de política.
report-to	Especifica un grupo (definido en la cabecera Report-To) al cuál el User-Agent envía los informes acerca de violación de políticas.

Un ejemplo de uso podría ser:

Content-Security-Policy: script-src 'self'

Se considera una política a un conjunto de directivas con un valor determinado, mientras que una directiva es una pareja "nombre/valor" acorde a su funcionalidad. Por el contrario, una violación se considera cualquier acción o recurso que va en contra de alguna de las directivas de una política. En caso de que así se especifique, las violaciones producidas serán reportadas en los lugares que se habiliten para ello.

5.7 X-Permitted-Cross-Domain-Policies

Un fichero de política Cross-Domain es un documento XML que concede a un cliente web, como pudiera ser Adobe Flash Player o Adobe Acrobat, permiso para manejar datos entre dominios distintos. Cuando el cliente hace una petición de contenido alojado en un dominio concreto y ese contenido hace peticiones a través de un dominio distinto del que al que pertenece, el dominio remoto necesita un fichero de política Cross-Domain para poder tener acceso al dominio origen y así poder completar la transacción del cliente.

Los ficheros de política conceden acceso a los datos del dominio y también conceden permiso para conexiones basadas en sockets. La ubicación más común para un fichero de política en un servidor

es el directorio raíz con el nombre crossdomain.xml. Los ficheros de política que se almacenan así reciben el nombre de Ficheros Maestros de Política (master policy files).

Los valores que puede tomar son:

Valor	Descripción
none	No se permiten ficheros de políticas para ninguna parte del servidor objetivo, ni siquiera el conocido como "master policy file".
master-only	Solo se permite el "master policy file".
by-content-type	[Solo HTTP/HTTPS] Solo los ficheros de política con Content-Type: text/x-cross-domain-policy están permitidos.
by-ftp-filename	[Solo FTP] Solo los ficheros de política con nombre crossdomain.xml (por ejemplo URLs que acaban en /crossdomain.xml) están permitidos. .
all	Todos los ficheros de política están permitidos en este dominio objetivo.

Un ejemplo de uso puede ser:

X-Permitted-Cross-Domain-Policies: none

5.8 Referrer-Policy

La cabecera HTTP de Referrer-Policy indica qué información de referencia (enviada en el Referer) se debe incluir en las peticiones que se lleven a cabo. Los posibles valores que puede tomar son:

Valor	Descripción
no-referrer	El Referer será omitido completamente. Ningún tipo de información relacionada con el Referer se envía con las peticiones HTTP.
no-referrer-when-downgrade	Comportamiento por defecto del User-Agent si no se especifica ninguna política. El origen se incluye como referer si se envía a un destino seguro (HTTPS->HTTPS), pero no se incluye si se envía a un destino menos seguro (HTTPS->HTTP).
origin	Solo envía el origen del documento en cualquier caso. Si el documento es https://example.com/page.html se enviará como Referer https://example.com/.
origin-when-cross-origin	Envía la URL completa cuando la petición va al mismo dominio origen, y solo envía el origen en el resto de casos.

same-origin	El Referer solo se enviará a destinos dentro del mismo dominio, cuando sea a dominios distintos no se incluirá el Referer.
strict-origin	Solo envía el origen del documento como Referer si se envía a un destino seguro (HTTPS->HTTPS), pero no se incluye si se envía a un destino menos seguro (HTTPS->HTTP).
strict-origin-when-cross-origin	Envía la URL completa si la petición es al mismo origen, solo envía el origen del documento a un destino seguro y distinto del origen (HTTPS->HTTPS), y no envía nada a destinos no seguros (HTTPS->HTTP).
unsafe-url	Envía la URL completa (delimitada entre paréntesis) si la petición es al mismo origen.

La idea tras este exhaustivo control de la información que se comparte en el Referer es fundamentalmente la de controlar la Privacidad y Seguridad del usuario. Tiene especialmente sentido cuando el usuario se encuentra en páginas con perfiles o sesiones personales y al acceder a enlaces puede filtrarse parte de esa información a través del Referer.

Un ejemplo de esta cabecera puede ser:

Referrer-Policy: no-referrer

6 CONCLUSIONES

A lo largo de este documento se ha explicado en qué consiste el OWASP Secure Headers Project, que se centra en dar a conocer a la comunidad ocho cabeceras del protocolo HTTP que ayudan a mejorar la seguridad en las comunicaciones web entre cliente y servidor. El uso de dichas cabeceras es fomentado por la Fundación OWASP como forma de proteger las aplicaciones web frente a vulnerabilidades extensamente conocidas y fácilmente evitables, cubriendo un espectro de riesgos de seguridad web bastante amplio.

La responsabilidad de ofrecer un entorno de comunicación web más seguro requiere de la implicación de los desarrolladores, administradores de sistemas, fabricantes e incluso del propio usuario final, de manera que toda la comunidad contribuya a mantener y ampliar los mecanismos de seguridad en las comunicaciones web.

7 GLOSARIO

Clickjacking (Secuestro de clic) : técnica maliciosa para engañar al usuario con el fin de obtener información acerca del mismo o buscando tomar el control del ordenador. Explota vulnerabilidades en los navegadores tomando forma de código embebido o de algún script en la página web mostrada y se ejecuta sin que el usuario tenga conocimiento.

Content Sniffing: mecanismo usado por los navegadores que consiste en inspeccionar un flujo de datos para deducir el tipo de formato de fichero que se está tratando. Se recurre a esta práctica cuando los metadatos recibidos son insuficientes para decidir el tipo de fichero que se está tratando en cada momento y cómo será representado por parte del navegador.

Cross-Site Scripting:: es un tipo de vulnerabilidad típico de las aplicaciones web que permite a una tercera persona inyectar, en páginas web visitadas por el usuario, código JavaScript o en otro lenguaje similar.

Downgrade: ataque que fuerza a pasar de un modo de operación seguro (por ejemplo, comunicación encriptada) a uno inseguro. Fue una vulnerabilidad descubierta en OpenSSL y permitía al atacante negociar una comunicación usando una versión de TLS más antigua.

Frame (Marcos HTML**):** los marcos HTML permiten mostrar páginas con vistas múltiples, ya sea a través de diferentes ventanas o subventanas independientes. Con el uso de las vistas múltiples se puede mantener cierta información visible mientras otras vistas se desplazan o se sustituyen. Por ejemplo, dentro de una misma ventana, un marco podría mostrar un gráfico estático, un segundo marco un menú de navegación, y un tercero el documento principal que puede ser desplazado, o reemplazado al navegar por el segundo marco.

Man-in-the-Middle: es un ataque con el que se consigue la capacidad de leer, insertar y modificar a voluntad mensajes entre dos partes sin que ninguna de ellas sea consciente que el enlace de comunicación ha sido violado. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas.

Session Hijacking (Cookie Hijacking): ataque fundamentado en el uso de una sesión válida para lograr acceso no autorizado a información o servicios en un sistema informático. El éxito de este ataque se basa en robar una cookie usada para identificar a un cliente en un servidor, y de esa manera hacerse pasar por cierto cliente con sesión abierta cuando no es el caso.

8 DOCUMENTACION DE REFERENCIA

https://www.owasp.org/index.php/OWASP_Secure_Headers_Project
https://tools.ietf.org/html/rfc6797
https://www.owasp.org/index.php/HTTP_Strict_Transport_Security
https://www.owasp.org/index.php/Test_HTTP_Strict_Transport_Security_(OTG-CONFIG-007)
https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
https://www.chromium.org/hsts
https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security
https://raymii.org/s/tutorials/HTTP_Strict_Transport_Security_for_Apache_NGINX_and_Lighttpd.html

https://tools.ietf.org/html/rfc7469

https://tools.ietf.org/html/draft-ietf-websec-x-frame-options-01

https://www.owasp.org/index.php/Clickjacking

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.virtuesecurity.com/blog/understanding-xss-auditor/

https://www.veracode.com/blog/2014/03/guidelines-for-setting-security-headers
http://zinoui.com/blog/security-http-headers#x-xss-protection

https://msdn.microsoft.com/en-us/library/gg622941%28v=vs.85%29.aspx

https://blogs.msdn.microsoft.com/ie/2008/09/02/ie8-security-part-vi-beta-2-update/https://www.w3.org/TR/CSP/

https://developer.mozilla.org/en-US/docs/Web/Security/CSP
https://www.owasp.org/index.php/Content_Security_Policy
https://scotthelme.co.uk/content-security-policy-an-introduction/

https://www.adobe.com/devnet/adobe-media-server/articles/cross-domain-xml-for-streaming.htm

https://www.w3.org/TR/referrer-policy/
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy