JaSiLDBG

JavaScript inLine Debugger

The handbook of the fastest debugger.

Por: Sirdarckcat y Crack_X
elhacker.net

Índice:

Introducción:

JavaScript inLine Debugger es una herramienta, que todos los usuarios pueden usar, independientemente del navegador. Esta herramienta nos permite ejecutar javascript arriba de la web sin tener que recargar, usar proxy, ni guardarla.

Esta herramienta no es algo nuevo, pero es poco conocida, poco comentada y muy potente, el objetivo de este trabajo es dar a conocer sus capacidades, y mostrar la facilidad de su uso.

Para poder utilizar esta herramienta, es necesario tener conocimiento muy básico de Programación Orientada a Objetos (OOP) y conocer un poco sobre el standard HTML de W3C.

Algunas de las características de este debugger son:

- No necesita instalación, solo un teclado y saber javascript.
- Capacidad de modificar cookies.
- Capacidad de modificar valores de campos de formularios, tanto visibles como ocultos, deshabilitar maxLength y permitir su edición.
- Capacidad de anular scripts que no permiten el ver el código fuente o seleccionar texto.
- Capacidad de mostrar el código de funciones encriptadas o escondidas.
- Capacidad de modificar variables y funciones, así como mostrar todas las propiedades de estos.
- Capacidad de emular eventos y establecer / modificar actuales.
- Capacidad de modificar el código fuente del programa, o solo una sección del mismo.

La manera de hacer esto es usando el protocolo *javascript:* disponible en:

- Internet Explorer
- Mozilla Firefox
- Opera
- Otros..

Para saber si tu navegador soporta este método, escribe en la barra de direcciones:
`javascript:alert("Hola Mundo");`
Si ves una ventana con "*Hola Mundo"*, tu navegador es compatible.

Los protocolos que pueden usarse para el *JaSiLDBG* son:

- *javascript:*
- *vbscript:* (La sintaxis es diferente)
- *jscript:* (Muy Inestable, ni lo intenten)

Para el control de errores, en firefox, debes ver la consola JavaScript.

Objetos

Los objetos son muy importantes en javascript al momento de modificar valores o llamar a funciones, estos son un conjunto de propiedades, atributos, funciones de algo en particular, por ejemplo, supongamos que tu monitor es un objeto de javascript.
Los atributos que tiene serian:
<MONITOR>
- tamaño=75x75;
- tipo=pantalla_plana;
- peso=15Kg;
- marca=Acer;

Sus funciones (en mi monitor equivalen a botones..) serian:

- apagar/encender();
- brillo();
- contraste();
- incrementar();
- decrementar();

y sus propiedades serian:

- color;
- posición;
- orientación;

la diferencia entre propiedades y atributos, es que los atributos, no los puedes cambiar, y las propiedades si, por ejemplo:

window.screen.availwidth no puedes modificarlo.
window.title si puedes modificarlo
window.open() abre una nueva ventana, ejecuta una acción es una función.

Para manejar el explorador, se usan 2 objetos predefinidos en el motor javascript, que son:
1.- ***document***
se refiere al documento, este tiene todos los objetos del documento HTML, body, tablas, divs, etc.. aunque realmente document es de window, es importante definir sus diferencias.
2.- ***window***
se refiere a la ventana del explorador, este contiene la mayoría de las funciones y propiedades del explorador.

El objeto document tiene todos lo que conforma el documento HTML, desde el

<body> hasta un simple <a href="">, mas adelante veremos como obtener control sobre ellos.

Realizar acciones:

Hay tres maneras distintas de realizar acciones en el JaSiLDBG, y dependen de que es lo que quieras, hacer, por ejemplo, si solo quieres modificar alguna propiedad, no es necesario ver el valor de retorno, si solo quieres ver el valor de una variable o propiedad no es necesario ejecutar una acción, etc..

1. **void(ACCIÓN);**
   Simplemente ejecuta la acción, por ejemplo:

   ```
   javascript:void(document.title='JaSiDBG');
   ```

2. **alert(VALOR/ACCIÓN);**
   Muestra el valor de retorno de cierta acción, propiedad o valor, por ejemplo:

   ```
   javascript:alert(window.screen.availWidth);
   javascript:alert(unescape("%2521"));
   ```

3. **void(prompt(null,VALOR/ACCION));**
   Permite seleccionar y copiar cierto valor, por ejemplo:

   ```
   javascript:void(prompt(null,document.cookie));
   ```

Este ultimo, permite al usuario copiar al portapapeles alguna propiedad, la caja de texto que muestra el prompt, es capaz de mostrar los caracteres de salto de línea, por lo que podrás copiar el texto normalmente y al pegarlo se hará exactamente como esta en el documento.

Además si no necesitamos valor de retorno, solo debemos colocar void(); al final de la función, recordemos que puedes colocar, mas de 1 comando por ves, separando cada comando con ; (punto y coma), o agregando %0d%0a (significa salto de linea).

Obtener y modificar COOKIES

Las cookies se encuentran en la propiedad "*cookie*" de "*document*":

```
document.cookie;
```

Puedes modificar y crear nuevas.

Este script creara una cookie llamada CDP

```
javascript:void(document.cookie="CDP=12345678901234567890123456789012");
```

Funciona de la siguiente manera:

```
void(document.cookie="CDP=12345678901234567890123456789012");
```

- ***void();*** - Indica que solo queremos ejecutar la acción.
- ***document.cookie***= - Indica que vamos a modificar la cookie.
- "***CDP***= - Indica que la cookie llamada CDP sera modificada
- ***12345678901234567890123456789012"*** - Indica el valor de la cookie CDP.

Este script nos mostrara las cookies.

```
javascript:alert(document.cookie);
```

El siguiente script, nos permitirá modificar cada uno de los valores de las cookies individualmente.

```
javascript:for(var ca in
document.cookie.split(";")){document.cookie=prompt("Nuevo valor de
cookie",document.cookie.split(";")[ca])}alert("Las cookies quedaron
asi:\n"+document.cookie.split(";").join("\n"));
```

Obtener OBJETO:

Existen múltiples formas de obtener un objeto del documento, usando funciones nativas de javascript, (en Mozilla Firefox puede usar el DOM Inspector (vea referencias)):

1. **document.getElementById(ID);**
   Al crear un objeto en HTML se puede especificar un campo llamado id, este debe ser único en el documento, y sirve para identificarlo respecto a los demás, al usar la función document.getElementById(); obtendrás como resultado el objeto cuyo Id corresponde al especificado.
   por ejemplo:

   ```
   javascript:alert(document.getElementById("SOME"));
   ```

   este regresara el objeto cuyo id es "SOME", si no existe, regresara "null", es decir, nada.

2. **document.getElementsByName(NOMBRE)[];**
   Regresa un Array con los objetos con esas caracteristicas.
   A diferencia del anterior, varios objetos pueden tener el mismo nombre, es decir, nos regresara en cada elemento del Array un objeto, por ejemplo:

   ```
   javascript:alert(document.getElementsByName("usuario").length);
   ```

eso te dirá cuantos objetos tienen de nombre "usuario". Para un simple ejemplo entremos www.google.com y pongamos:

javascript:alert(document.getElementsByName("btnG")[0].value="Yo Busco!");

Veremos que el botón llamado *btnG* ahora dice "Yo Busco!" , le hemos cambiado el valor .

3. **document.getElementsByTagName(TAG)[];**
Regresa un Array con los objetos con esas características.

Este quizá, es el mas útil, te regresa el objeto cuyo TAG sea igual al especificado, por ejemplo:

```
javascript:alert(document.getElementsByTagName("img")[0].src);
```

este te regresara la url de la primera imagen del documento.

4. Por grupos:

a.- document.frames

Este nos regresa un array con los frames que conforman el documento, este solo esta disponible en Internet Explorer.
Nos regresara en orden de aparición, los <iframe> y los <frame>, no haber ninguno, el valor será igual a null.

b.- document.forms

Este nos regresa un array con los formularios que hay en el documento, es importante aclarar, que para obtener el objeto de un elemento de un formulario, debemos acceder a él a través de su formulario, por ejemplo:

```
document.forms[0].usuario.value;
```

Donde usuario es el nombre o "name" del campo de formulario.

c.- document.links

Este nos regresara los vínculos del documento, en el 90% de los casos será igual a obtener document.getElementsByTagName("a"); es interesante aclarar, que en Internet Explorer puedes hacer uso de "click();" que simula un clic del mouse, sin embargo firefox no lo permite.

d.- document.layers

Presentado originalmente en netscape, y tras algunas adaptaciones en Internet Explorer es obtener los tags "<layer>" y " " su uso es muy extraño, pero puede servir para paginas que usen estos tags.

e.- document.applets

Para obtener control de applets de java, e interactuar con ellos, puedes obtener el objeto con document.applets idéntico a getElementsByTagName("applet");

5. **window.ID (SOLO IExplorer)**

   Microsoft Internet Explorer permite obtener objetos de una manera mas sencilla, solo usando el ID, sin embargo este método NO es exportable y no lo recomiendo.

Obtener PROPIEDAD de OBJETO:

Simplemente llámalo así:
OBJETO.PROPIEDAD;

Algunas propiedades modificables útiles que te pueden servir:

- **src**

  Este sirve para obtener el valor de "src" o source de una imagen, película, sonido, script, etc.. especialmente cuando se desactiva el "context menu" del documento. (el cual veremos mas adelante como re-habilitar), también puedes modificarlo.

- **href**

  Este nos regresara, si es un hipervínculo a donde apunta, útil en "document.links", también puedes modificarlo.

- **value**

  Nos regresara el valor de un campo de formulario, y podremos cambiarlo a nuestro antojo, es útil por ejemplo para descubrir el valor de un campo de formulario tipo hidden y modificarlo, de esta manera, por ejemplo:

```
javascript:alert(document.forms[0].price.value=0);
```

- **length**

En caso de ser un array o un objeto, nos regresara la cantidad de propiedades que tiene, en caso de ser una cadena de texto/String nos regresara la longitud del mismo.

- **maxLength**

  Una de las protecciones mas molestas en el momento de hacer un ataque, sin tener que modificar el archivo ni modificar cabeceras, solo anulamos el valor maxLength igualándolo a "null", o poniéndole un valor extremadamente alto, o "-1".

- **readOnly**

  Lo crean o no, muchas veces cuando no quieren poner datos ocultos, lo ponen con la opción readOnly o disabled, estos con la misma facilidad pueden ser desactivados, solo iguálenlos a "false".

- **disabled**

  Igual que readOnly, solo es necesario igualarlo a 0 o a false, y la edición de ese campo de formulario estará habilitada.

- ***Acciones dentro de formularios.***

  - **focus()**

    Para ayudarte a detectar algún campo que no encuentres o estés seguro cual es, la función focus() pondrá el cursor en el objeto especificado.

  - **click()**

    Solo para InternetExplorer, simula un clic, sin embargo en Firefox puedes emular un evento, como veremos mas adelante.

  - **select()**

    Funciona de manera similar a select, sin embargo es útil al momento de trabajar con campos de formulario de menús, por ejemplo un campo tipo "radio" se seleccionara al activar esta función, o un <option> tomara el valor del select.

Además, podemos hacer que una serie de comandos sean para las propiedades y atributos de un objeto, si colocamos a estas acciones dentro de un with(), es decir:

```
javascript:with(document){alert(title);alert(location.href);}
```

Nos mostrara el titulo y la URL del documento.

Sobre el objeto document

Este contiene algunas otras propiedades muy interesantes, que nos pueden servir, como:

- **document.scripts**
  Regresa los scripts del documento.
- **document.embeds**
  Regresa los objetos embed, como videos, objetos flash, etc..
- **document.mimeType**
  Regresa el mimeType del documento.. normalmente es *HTML Document*.
- **document.protocol**
  Regresa el protocolo del documento.
- **document.URL**
  Regresan la URL.. también puedes obtener la url sin urlencode en *document.URLUnencoded*.
- **document.images**
  Regresa las imágenes del documento.
- **document.anchors**
  Regresa las secciones del documento.
- **document.selection**
  Regresa lo que este seleccionado en el documento.
- **document.security**
  Si es HTTPS regresara si tiene un certificado válido.

Algunas de estas propiedades pueden no estar disponibles en su navegador.

document.location

Algunas otras propiedades que pueden usarse, para el control de la línea de tiempo del navegador, son los siguientes:

- **document.location.href** - te dice la url en la que esta el documento.
- **document.location.reload()** - actualiza la pagina, es equivalente al boton "refresh" o "actualizar".
- **document.location.hostname** – devuelve el nombre del host.
- **document.location.host** – devuleve el host..
- **document.location.hash** – devuelve la sección de anchor de la localización del documento, es decir lo que esta después de #.

- **document.location.search** – devuelve lo que esta después de ? y antes de #, en la url, es decir, las variables pasadas en método GET.
- **document.location.port** – devuelve el puerto de la url.. sin embargo, necesita ser especificado en la URL. Se necesita que el puerto este especificado, como este: *protocolo://hostname.host.domain:puerto/path?var=val#anchor*
- **document.location.pathname** – te regresa el path al archivo.
- **document.location.protocol** – regresa la sección del protocolo del documento, http: ftp: file: etc..
- **document.domain** - te dice en el contexto de que dominio se esta ejecutando el documento, algunas funciones no tienen acceso si el dominio del documento destino es diferente al original.
- **document.referrer** - te dice el referrer de ese documento.
- **window.history**
    - **length**

      Te regresa la cantidad de paginas guardadas en el historial de la sesión actual.

    - **go(int);**

      Te lleva a la posición en el historial especificada.

    - **back(int);**

      Te lleva atrás la cantidad de veces especificada.

    - **forward(int);**

      Te lleva adelante la cantidad de veces especificada.

Nota:
Los frames tienen un documento separado al original, por lo que sus valores pueden cambiar.

Obtener todas las propiedades de un objeto.

Algunas veces necesitas saber las propiedades de algún objeto.. para eso puedes usar la siguiente sintaxis:

```
for(var x in OBJETO){
   x=PROPIEDAD_DE_OBJETO;
}
```

por ejemplo:

```
javascript:var m = "[-] document:";for(var x in document){m+="\n|-
"+x}alert(m);
```

Este funciona de manera similar al foreach de otros lenguajes.

También vale mencionar, que para acceder a la propiedad del objeto accedes como si fuera un arreglo.. es decir, para obtener el valor de las propiedades seria así:

```
javascript:var m = "[-] document:";for(var x in document){m+="\n|-
"+x+" = "+document[x]}alert(m);
```

Acceder a OBJETOS por nivel

Se usan los siguientes atributos:

1.- ***self***
se refiere al documento actual
2.- ***top***
se refiere al documento principal
3.- ***parent***
se refiere al objeto padre

Reemplazar valores.

Si queremos modificar valores en el código fuente, o alguna propiedad, de forma recurrente, es decir no hacerlo uno por uno..

Lo mas optimo es usando split() y join(), por ejemplo:

```
javascript:document.documentElement.innerHTML.split("a").join("A");
```

También puedes usar la función replace();

```
javascript:document.documentElement.innerHTML.replace("a").join("A");
```

esto cambiara todas las "a" por "A".
es usado generalmente para anular algunos scripts:

```
javascript:document.documentElement.innerHTML.split("script").join("xmp")
;
```

También puedes usar expresiones regulares en javascript:

```
javascript:document.documentElement.innerHTML.split(/(S|s)(C|c)(R|r)(I|
i)(P|p)(T|t)/).join("xmp");
```

La función replace(); también acepta expresiones regulares.

Para mas información de expresiones regulares visitar:

http://es.wikipedia.org/wiki/Expresión_regular

Obtener/Crear Estilo:

Existen 2 maneras, con:
1.- ***currentStyle***
sintaxis:
OBJECTO.currentStyle[PROPIEDAD];

2.- ***getComputedStyle***
sintaxis:
```
 document.defaultView.getComputedStyle(OBJETO,null).getPropertyValue(PROP
IEDAD);
```

Podemos generar hojas de estilo con:

document.createStyleSheet();
y después acceder al código css con la propiedad cssText.

Obtener el tamaño y posición de un objeto:

1.- Usando las funciones ya mencionadas de Estilo
- style.top
- style.left
- style.height
- style.width

2.- Con las propiedades offset:

- *offsetHeight*
- *offsetLeft*
- *offsetTop*
- *offsetWidth*

Estas son usadas comúnmente para obtener el tamaño de la pantalla.

Obtener/Modificar el código fuente de objetos:

Existen 2 maneras casi idénticas para obtenerlo:
1.- *OBJETO****.innerHTML;***
Este obtiene el código HTML interno
2.- *OBJETO****.outerHTML;***

Este obtiene el código HTML que es generado por el código HTML interno.

De la misma forma puedes obtener el texto sin código HTML así:
1.- *OBJETO**.innerText;***
Este obtiene el texto del documento.
2.- *OBJETO**.outerText;***
Este ultimo obtiene el texto generado por el código HTML interno.

Agregar/Reemplazar/Quitar segmentos de código.

Con javascript podemos modificar solo las secciones deseadas de código, por ejemplo, quitar divs, agregarlos, reemplazarlos, clonarlos, etc..

En el siguiente ejemplo, colocaremos un campo de formulario al final del documento:

```
javascript:l=document.createElement("input");l.setAttribute("value","Hola");void(docume
nt.body.appendChild(l));
```

Aquí usamos 3 funciones, que explicare a continuación:

document.createElement
Esto genera un objeto HTML virtual.

setAttribute
Agrega un atributo a un objeto virtual.

appendChild
Esto agrega el objeto virtual, a un objeto real, y lo genera.

También hay otra funciones que te pueden servir como:

replaceChild(objeto1,objeto2);
Este reemplaza el objeto1 con el objeto2.

removeChild(objeto);
Este quita el objeto del documento.

cloneNode(bool);
Este clona un atributo, y lo guarda en una variable, como objeto virtual.

Obtener/Generar funciones dinámicamente.

Para obtener el código de una función de javascript, aunque esta este encriptada, puede ser obtenida de la siguiente forma:

alert(funcion);

La misma técnica puede ser usada para obtener “classes”

Normalmente hay funciones, muy molestas que o borran el contenido del portapapeles, o hacen alguna otra cosa que nos molesta, a la hora de analizar una página web.

Estas funciones pueden ser redeclaradas, para nuestra facilidad.

Simplemente hay que redeclararlas, por ejemplo, para deshabilitar el clic derecho, es común hacer algo así:

```
<script>
function blah(){
alert("Derechos Reservados");
return false;
}
document.oncontextmenu=blah;
</script>
```

Para obtener el código de la función solo haríamos:

```
javascript:alert(blah);
```

Y para volver a habilitarlo, solo ponemos:

```
javascript:function blah(){return true;}
```

o también podemos reasignar el evento, como veremos a continuación.

Antes, otra forma de declarar variables, es la siguiente:

```
javascript:var blah = new Function("return true;");
```

o de esta otra:

```
javascript:var blah = new function(){return true;}
```

Asumir control de eventos.

Asumir eventos significa que tomes control sobre algún evento, estos se manejan de forma diferente en Netscape, usando la función document.captureEvents(); sin

embargo, usaremos Mozilla FireFox y Internet Explorer , que usan los eventos como propiedades de un objeto..

Por ejemplo, window.onunload será llamado cuando se valla a cerrar la ventana.

document.oncontextmenu será llamado cuando se de clic derecho al documento.

Ahora, este es un programa que deshabilita el clic derecho, veremos como con nosotros asumir el evento document.oncontextmenu podemos evadir esta protección.

El código es el siguiente:

```
var message="Copyright privado.inc";

function clickIE()
{
  alert(message);

  if (document.all)
  {
          document.oncontextmenu=new Function("return false");
          return false;
  }else{
          document.oncontextmenu=new Function("return false");
          return document.oncontextmenu();
  }
}

document.oncontextmenu=clickIE;

var lWd = document;
with(lWd)oncontextmenu=new Function("return false");
```

Este código es considerablemente seguro, salta algunos plugins que re-habilitan el clic derecho.. pero no puede contra JaSiLDBG.

Nosotros lo único que debemos hacer es:

```
javascript:void(document.oncontextmenu = new Function("return true;");
```

También podemos asumir control sobre onunload, onbeforeunload, onmousedown, etc.. por ejemplo hay códigos que verifican si el carácter que escribiste en un campo de texto es válido, y si no lo es, no permite su captura.. aunque esto parece que va a cambiar en futuras versiones de javascript, por peligro a la privacidad de los usuarios.. aun así, estos códigos se deshabilitan modificando los eventos:

Onkeypress

Onkeydown

Onkeyup

Onchange

Como ya vimos, para obtener el objeto puede ser:

Supongamos que esta en un único form, y el campo se llama “usuario”

Es decir.. seria algo así:

```
javascript:void(document.forms[0].usuario.onkeypress=new function(){return true});
```

Para obtener todos los eventos de un documento, puedes ver en el standard de W3C o con las herramientas del JaSiLDBG, que encontraras al final del documento.

Detener/Crear Intervalos.

Otra técnica usada para “evitar” que copies código, o imprimas pantalla, es limpiando el portapapeles.. este código por ejemplo hace eso:

```
function clptmr()
{
  window.clipboardData.clearData();
}
var ClrClp = setInterval("clptmr()", 100);
```

Aquí, se genero un intervalo, para que cada 100 milisegundos, se borre el portapapeles.

Para limpiar el intervalo, usamos la función clearInterval();

Es decir, para el código de ejemplo, el Intervalo, se limpia solo con poner:

```
javascript:void(window.clearInterval(ClrClp));
```

Además nosotros también podemos crear nuestros propios Intervalos, para que se ejecute algún comando cada cierto tiempo.

Esto se hace así:

```
javascript:void(window.setInterval("alert('Hola');",1000));
```

Para obtener el control sobre un intervalo que no fue guardado, se puede usar el siguiente código: javascript:void(clearInterval(setInterval("",0)-1));

Lo que hace, es obtener control sobre el último intervalo generado.
Si has seguido este documento desde el principio, también me podrías decir, que se puede deshabilitar el intervalo de ejemplo, rescribiendo la función clrtmr(), si lo pensaste, vamos bien ☺.

Obtener la fuente de imágenes, scripts, películas, etc..

Para obtener la url en donde esta una imagen, película, etc.. puede hacerlo:

Simplemente con:
*OBJETO.**src;***

Sin embargo con frames es así:
*OBJETO.**location;***

Y para links, entre otros (si quiere obtener el anchor puede usar OBJETO.**hash**):
*OBJETO.**href***

Incluso con frames, o con cualquier documento, puedes obtener el referrer:
*OBJETO.**document.referrer;***

Por ejemplo, para obtener la fuente de una imagen, puedes usar:

```
javascript:alert(document.getElementsByTagName('img')[0].src);
```

que obtendría la fuente de la primera imagen del documento.

Obtener el documento completo como objeto.

Existen 2 maneras, usen la que quieran:
1.- *document.getElementsByTagName("html")[0];*
2.- *document.documentElement;*

Esto te puede servir para obtener el código fuente real del documento, de la siguiente manera:

```
javascript:alert(document.documentElement.innerHTML);
```

Trabajando con objetos de Shockwave Flash

Flash, puede interactuar con javascript y viceversa, aunque la comunicación javascript=>flash es mas complicada que actionscript=>javascript, es posible. algunos de las funciones que el control ActiveX de Flash y LCn de Mozilla son:

Control de pelicula:

1. ***IsPlaying();***
   Estado de ejecución de la película.
2. ***Play();***
   Inicia la película.
3. ***Stop();***
   Detiene la película.
4. ***Back();***
   Regresa un frame la película.
5. ***Forward();***
   Adelanta un frame la película.
6. ***Rewind();***
   Regresa la película al primer frame.
7. ***GotoFrame(int);***
   Manda la película al frame especificado.
8. ***FrameLoaded(int);***
   Regresa true si la película esta en la frame especificada.
9. ***CurrentFrame();***
   Regresa la frame actual.

Control de reproductor:

1. ***FlashVersion();***
   Regresa la versión de flash en ejecución.
2. ***LoadMovie(int,URL);***
   Carga cierta película *(int)* de la *URL* especificada.
3. ***PercentLoaded();***
   Te dice el porcentaje de carga de la película.
4. ***SetZoomRect(IZQUIERDA,ARRIBA,DERECHA,ABAJO);***
   Establece el zoom en la posición indicada.

Control de programa y propiedades:

1. ***SetVariable(VARIABLE,VALOR);***
   Introduce en la *VARIABLE* en *VALOR* especificado.

2. ***GetVariable(VARIABLE);***
   Regresa el valor de la *VARIABLE*.
3. ***TCurrentFrame(INST);***
   Regresa la frame de cierta instancia.
4. ***TCurrentLabel(INST);***
   Regresa el nombre del frame.
5. ***TCallFrame(INST,int);***
   Llama a la función que hay en esa frame.
6. ***TCallFrame(INST,NOMBRE);***
   Llama a la función que hay en la frame con ese nombre.
7. ***TGetProperty(INST,PROPIEDAD);***
   Regresa la *PROPIEDAD* de la instancia.
8. **TSetProperty(INST,PROPIEDAD,VALOR);**
   Establece cierto VALOR a la PROPIEDAD de la instancia.

Acceder a una variable/objeto

Si el identificador del objeto es: "camino", se accede a él de la siguiente manera:

/:camino

es decir, si quisiéramos cambiar de posición al objeto "camino", simplemente seria algo como:

```
javascript:void(document.getElementsByTagName("embed")[0].TSetProperty("/
:camino",0,100));
```

para cambiar el contenido del alguna variable u objeto, seria algo asi:

```
javascript:void(document.getElementsByTagName("embed")[0].SetVariable("/:
camino:variableopropiedad","valor"));
```

Descubrir variables desconocidas

El "demo" del programa "Flash Decompiler" de Eltima software, te permite ver el interior de los SWF, así como su código de action script, lo pueden descargar gratuitamente en: **http://www.flash-decompiler.com/**.

Vea las referencias

Otras funciones y métodos para usar con flash, pueden ser encontradas en el sitio web de soporte de flash.

También puede hallar ejemplos del uso de JaSiLDBG, con flash, en la página de "**Ejercicios de ejemplo para JaSiLDBG**".

Trabajando con controles ActiveX

Los controles ActiveX de Microsoft pueden interactuar con javascript, el objeto OCX que crees debe tener las
funciones y propiedades en modo "Public", son llamadas de la siguiente manera:

*OBJETO**.funcion();***
*OBJETO**.propiedad;***

OBJETO debe apuntar al <OBJECT> que controla el objeto ActiveX, es importante aclarar que una gran cantidad de bugs en controles ActiveX, son porque el programador cree, que es imposible llamar funciones del mismo dominio y sin romper firmas de integridad del código. Esto es obviamente falso con JaSiLDBG, por ejemplo por medio de JaSiLDBG, podemos abrir ventanas de conversación en MSN Messenger desde la pagina del hotmail, podemos iniciar Juegos en el mismo, así como en webs como "www.gamedesire.com" tenemos la capacidad de modificar propiedades que se supone son seguras, también valores en webs de casinos, que guardan en un control ActiveX algunos valores esenciales (según para mas seguridad), que modificándolos puedes comprometer el Casino, y que creen que por usar SSL están libres de esto (al igual que algunos bancos). Tristemente no es así, al menos con JaSiLDBG las capacidades son mucho mas desarrollables en el campo de controles ActiveX, sin embargo por la cantidad tan larga de controles que hay nos es imposible enumerarlas todas, además de que su explotación es muy fácil, una ves entendido el concepto.

Modificando y obteniendo Headers con javascript y XMLHTTP.

Dependiendo de tu navegador los objetos que manejan XMLHTTP son diferentes. Para Opera y Firefox:

var cx = new *XMLHttpRequest();*

para IEplorer:

var cx = new *ActiveXObject("Microsoft.XMLHTTP");*
ó
var cx = new *ActiveXObject("Msxml2.XMLHTTP");*

Ya que tenemos el objeto, podemos por ejemplo, hacer una petición a la web actual.

(SOLO PUEDES HACER PETICIONES DE AJAX AL MISMO DOMINIO EN EL QUE ESTAS)

```
javascript:var cx = new
XMLHttpRequest();cx.open("GET","/",false);cx.send(null);alert(cx.re
sponseText);
```

Este es:

1. Establecemos que la variable "cx" tendrá un objeto XMLHTTP.
2. Abrimos en método "*GET* " el directorio actual.
   el false es para que el documento espere hasta que este listo.
3. Enviamos la petición.
4. Y mostramos el texto que nos respondió la petición.

Hay mas métodos, en lugar de GET, por ejemplo:

- **POST,** y la información POST la envías en la función send("INFO=AQUI&MAS=MAS");
  es importante aclarar que para que funcione, y sea reconocido por el servidor la petición POST, debes agregar la cabecera, "Content-Type" con "application/x-www-form-urlencoded".
- **HEAD**, que solo regresa la cabecera.
- **SEARCH**, no disponible en todos.
- **TRACE**, desactivada en las ultimas versiones de XMLHTTP.. nos regresaba la petición que nosotros mandamos al servidor.. el uso de esta cabecera e conjunto con ataques de XSS (Cross Site Scripting), da lugar a un ataque llamado XST, (Cross Site Tracing).

y todos los que soporte tu servidor.. debes ponerlo en Mayúsculas.
para mandar una cabecera personalizada, es así:

cx.setRequestHeader("Referrer: http://www.hey.com/");

despues de *open()* y antes de *send();*
para recibir todas las cabeceras que respondió el servidor:

cx.getAllResponseHeader();

después de *send();*
para recibir solo una, por ejemplo:

cx.getResponseHeader("Content-type");

después de *send();*

También es importante aclarar, que puedes ejecutar comandos uno por uno.. es decir no debe de ser todo de una vez.. por ejemplo:

En la web actual coloca esto:

```
javascript:void(cx = new ActiveXObject("Microsoft.XMLHTTP"));
```

después:

```
javascript:cx.open("POST","/",false);cx.setRequestHeader('Content-
Type','application/x-www-form-
urlencoded');cx.send("x=1&w=2");alert(cx.responseText);
```

y después puedes usar cx como el objeto sin tener que crear uno cada vez.

Otras utilidades de JaSiLDBG.

En nuestra experiencia, hemos descubierto, que JaSiLDBG, puede servir para muchas cosas mas de las que hemos hablado.. sin embargo, dado que las capacidades son muchas, solo mencionaremos algunas ocasiones en las que podría servir.

Calculadora

Simplemente usando la librería Math, o usando simples funciones, tanto como suma resta, multiplicación, etc.. como cómputos modulares, y binarios, también obtener valor de constantes, como PI o E, o crear números aleatorios, o redondeo, etc.. una lista de todas las funciones de Math, puede ser encontrada a continuación:

http://www.devguru.com/technologies/ecmascript/QuickRef/math.html

Criptoanálisis

Últimamente, hemos usado JaSiLDBG para criptografía, por ejemplo, Crack_X ha hecho una utilidad llamada Pescadito, que encripta texto seleccionado en diferentes algoritmos, como blowfish twofish, AES, o crea checksums como MD5 o SHA-1, y Sirdarckcat, gracias a JaSiLDBG, logro romper con éxito el algoritmo de encriptación de una página de puntuación.

Una función muy útil, para este tema es:

```
javascript:alert("aWq".charCodeAt(0));
```

charCodeAt(int);

regresara el valor ASCII de el carácter en el lugar seleccionado.

Entre otras tenemos las operaciones matemáticas, como XOR, que se interpreta con el símbolo ^, AND que se interpreta con &, OR que se interpreta con |, o para criptografía modular, %.

Quitar publicidad

Muchas veces publicidad, engañosa, o solo molesta no nos permiten visualizar muchas páginas web, o nos obligan a descargar un control ActiveX de gran peligrosidad, y que sin el, no podríamos descargar el archivo deseado, o

páginas que al cerrarse, se vuelven a abrir.. esto es muy común, y asumiendo eventos, como “onUnload” o “onBeforeUnload” podemos detener estos bucles.. también podemos ocultar divs, usando la propiedad:

Objeto.style.visibility=”Hidden”;

Depurar navegadores

Durante la elaboración de este documento, se descubrieron 2 bugs en Microsoft Internet Explorer, de NULL pointer exception, que podrían terminar en Buffer Overflow, y consecuentemente en ejecución de código remoto.

Crear expresiones regulares

JavaScript, tiene una función llamada RegExp, que permite la creación de expresiones regulares, también existe document.createExpression, un ejemplo y explicación de su uso, se encuentra a continuación:

http://www.devguru.com/technologies/ecmascript/QuickRef/regexp.html

Modificar páginas web con un solo clic

Usando comandos de JaSiLDBG en un bookmark, podemos modificar el aspecto de una web, con un solo clic. Wikipedia, por ejemplo, para agregar mas interactividad en la creación de artículos, ha promovido librerías para agregar opciones.. Además a lo largo del tiempo se han usado comandos en bookmarks para evadir el pago de Meegos animados de Microsoft © o para saltar la verificación de software genuino de Windows©, entre otros.

Automatizar acciones

Simplemente como macro, para generar algún loop, desde un simple flood, hasta generar un código de fuerza bruta.. o filtrar algún tipo de datos.

Explotar bugs de XSS, SQLi, RFI, etc..

Lo dejo al final, pero es el mas importante, el uso de valores ocultos, o cookies, que como ya vimos puede ser modificado, contiene información, que creen que con tener sesiones, control de referrer, es infalsificable, pero con JaSiLDBG, no es verdad..

Librería para JaSiLDBG, estigma.js

Hemos desarrollado una pequeña utilidad, con una serie de funciones, que te podría ayudar, para, desde investigar cabeceras con las que responde un servidor, editar el documento, como si fuera frontpage, o dreamwaver, usar algunos algoritmos de criptografía, como AES, Blowfish, también podrían hacer md5 checksums, o convertir campos de password, o escondidos en tipo texto.. etc..

Solo coloca el siguiente script en tu barra de navegador:

javascript:var est = document.createElement("script");est.setAttribute("src","http://sirdarckcat.googlepages.com/estigma.js");void(document.getElementsByTagName("head")[0].appendChild(est));

Sin embargo te recomiendo agregarlo como bookmark.

La lista de funciones se presenta a continuación:

resHeaders()
Regresará las cabeceras que son respondidas a una petición GET Standard, al documento actual.

fleHeaders(*file*)
Regresará las cabeceras que son respondidas a una petición GET Standard, al archivo especificado.

resSource()
Regresará el código fuente del documento actual.

fleSource()
Regresará el código fuente del archivo especificado.

emReq()
Regresará un handler de XMLHTTP, al documento actual.

explain()
Regresara un string con las propiedades y sus valores de un objeto.

moCookies()
Regresará una serie de diálogos para modificar individualmente las cookies.

geEvents()
Regresará un string con los eventos que tiene un objeto.

geGET()
Regresará, un array con las variables enviadas en método GET.

geCookies()
Regresará, un array con las variables en las cookies.

page(*obj*,*num*,*sort*)
Regresará, un string, que ordena los elementos de un array.

xpage(*obj*)
Regresará, una serie de diálogos con las propiedades de un array.

LoadLib(*url*)
Cargará la librería de funciones especificada al documento.

Pescadito()
Cargará las funciones de criptografía de Pescadito, de Crack_X

disMax()
Desactivara la limitación de máximo número de caracteres, en todos los campos de formulario.

disPass()
Convertirá todos los campos de formulario tipo password, a tipo texto.

disHide()
Hará visibles todos los campos de formulario tipo HIDDEN.

disFreeze()
Habilitará todos los campos de formulario, con la propiedad readOnly o disabled.

disAll()
Ejecutará, las funciones disMax(), disPass(), disHide(), disFreeze().

geCSS()
Regresará las hojas de estilo, que hay en el documento.

geScripts()
Regresará los scripts que hay en el documento.

EditHTML()
Cargará al final del documento, un iframe con WYSIWYG, para editar el cuerpo del documento.

ApplyHTML()
Una vez que hayas terminado de editar el documento con la función EditHTML(), esta función aplicara los cambios.

ASCII()
Regresará el valor ASCII de cada character de un string.

dec2hex()
Convertirá el número, en un string que contiene el valor hexadecimal de este número.

dec2bin()
Convertirá el número, en un string que contiene el valor binario de este número.

hex2dec()
Convertirá un string que contiene el valor hexadecimal de un número en su valor decimal.

hex2bin()
Convertirá un string que contiene el valor hexadecimal de un string con su valor binario.

bin2dec()
Convertirá un string que contiene el valor binario de un número, y lo convierte en decimal.

bin2hex()
Convertirá un string que contiene el valor binario de un número, y regresa otro string con su valor binario.

Otras funciones pueden haber sido agregadas, para ver las últimas funciones, entra a:

http://sirdarckcat.googlepages.com/estigma

Referencias

Referencia de JavaScript

- Mozilla
- DevGuru

DOM Inspector

- Mozilla

Referencia de HTML

- W3c

Referencia de Shockwave Flash

- Scripting with flash

Controles ActiveX

- Activando controles ActiveX

Ejercicios de ejemplo para JaSiLDBG

- http://www.elhacker.net/jasildbg/

Protocolo “javascript:”

- Microsoft Developer Network (MSDN)

Discusión de este documento

- **en *foro.elhacker.net***

Acerca del documento

Este documento fue desarrollado por medio de Zoho Writer, después se exporto a Word, y publicado en Google Docs.

Para la versión final, se generó un PDF, para la versión en ingles, y en español.

La última versión de JaSiLDBG puede ser vista en:
http://jasildbg.googlepages.com/es

Autores

Eduardo Vela (sirdarckcat)

Mail: sirdarckcat@gmail.com

Web: http://sirdarckcat.googlepages.com/

Yasser Hernandez (crack_x)

Mail: cybersurfer@gmail.com

Web: http://crackx.webcindario.com/

Licencia

Agradecimientos

Agradecemos al STAFF del foro de elhacker.net por darnos sus sugerencias, y apoyo, sobretodo a Alex Bove.