JaSiLDBG

JavaScript inLine Debugger

The handbook of the fastest debugger.

By: Sirdarckcat and Crack_X
elhacker.net

Index:

Introduction:

JavaScript inLine Debugger is a tool, which every user can use, independently from the browser you use. This tool allows us to execute JavaScript on top of a website without the need of either refreshing, using a proxy, neither saving it.

This tool isn't something new, but its not well known, not commentated much and very powerful, the objective of this job is get to know its capabilities, and demonstrate the easy use of it.

To be able to use this tool, it's necessary to have very basic knowledge of Object Oriented Programming (OOP) and to know a little bit about the HTML W3C standard.

A few characteristics about this debugger are:

- No need of installation, just a keyboard and to know JavaScript.
- Capability of modifying cookies.
- Capability of modifying forms values, visibles and hidden, disable maxLength and allow editing.
- Capability of disabling scripts that don't allow viewing the source code or selecting text.
- Capability of showing source code of encrypted or hidden functions.
- Capability of modifying variables and functions, as well as to show the properties of these.
- Capability to emulate events and establish / to modify presents.
- Capability of modifying the source code of the website, or just a section of it.

The way of doing this is by using the javascript: protocol available in:

- Internet Explorer
- Mozilla Firefox
- Opera
- Others...

In order to know if your browser supports this method, write in your address bar:
`javascript:alert("Hello World");`
If you see a message saying "Hello World", your browser is compatible.

The protocols which you can use with *JaSiLDBG* are:

- *javascript:*
- *vbscript:* (Syntax is different)
- *jscript:* (Very unstable, don't try it.)

To control errors, in firefox, use JavaScript Console (under tools).

Objects

Objects are very important in JavaScript when modifying values or calling functions, these are a set of properties, attributes, functions of something in particular, for example, suppose that your monitor is a JavaScript object. The attributes that it has would be:
<MONITOR>
-size=75x75;
- type=flat_screen;
- weight=15Kg;
- brand=Acer;

Its functions (in my monitor it's the same as the buttons..) would be:

- off/on();
- brightness ();
- contrast();
- increment ();
- decrement();

and its properties would be:

- color;
- position;
- orientation;

the difference between properties and attributes is that you can't change attributes, but properties you can; for example:

window.screen.availwidth cant modify it.
window.title can modify it
window.open() opens a new window, executes and action, it's a function.

To handle the explorer, we use 2 predefined objects in JavaScript motor, which are:
1.- ***document***
refers to the document, it has all the objects of the HTML document, body, tables, divs,etc... even dough document is from window, its important to define its differences.
2.- ***window***
refers to the window of the Explorer, contains most of the functions and properties of the Explorer.

The object document has everything that conforms an HTML, from the <body> to a simple <a href="">, later on you will learn how to obtain control over them.

Making Actions:

There are three different ways of realizing (making) actions in JaSiLDBG, and it depends on what you want to do, for example, if you only want to modify the value of some property, you don't need a return value, if you only whant to see the value of a variable or property then its not necessary to run an action, etc..

1. **void(ACTION);**
   It ejecutes an action, for example:

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

2. **alert(VALUE/ACTION);**
   Shows the return value of certain action, property or value, for example:

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

3. **void(prompt(null,VALOR/ACCION));**
   Allows to select and copy certain value, for example:

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

This last one allows the user to copy to the clipboard a property or the value returned; the textbox that shows the prompt is capable of showing the new line characters, thus, you could copy text normally so when you paste it would look exactly as it did in the document.

Also, if we don't need a return value, we only have to add a void(); at the end of the function. Remember that we can put more than one command per time, separating each command with a ; (semicolon), OR adding: %0D%0A (means new line).

Obtain and modify COOKIES

Cookies can be found in the property "cookie" of "document:

```
document.cookie;
```

You can modify and create new ones.

This script creates a cookie called CDP

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

This is how it works:

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

- ***void();*** - Indicates that we only want to execute the action.
- ***document.cookie***= - Indicates that we're modifying the cookie.

- "***CDP=*** - Indicates that the cookie named CDP will be modified.
- ***1234567890123456789012345678901 2"*** – Indicates the value of the cookie named CDP.

This script will show us the cookies:

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

The next script allows us to modify each value of the cookies, individually.

```
javascript:for(var ca in
document.cookie.split(";")){document.cookie=prompt("New value of
cookie",document.cookie.split(";")[ca])}alert("Cookies are like this
now:\n"+document.cookie.split(";").join("\n"));
```

Obtain the OBJECT:

Multiple forms exist of obtaining an object from the document, using native JavaScript functions (In Mozilla Firefox, you can use DOM Inspector (see references for more information)):

1. **document.getElementById(ID);**
   When you create an object in HTML you can specify a property called id, this must be unique in the document, and serves to identify it from the rest, when you use the document.getElementById(); function, you will obtain as a result the object which Id corresponds the one you specified.
   For example:

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

   This will return the object which id is "SOME", if it doesn't exist, returns "null", which means, nothing.

2. **document.getElementsByName(NAME)[];**
   Returns *an Array with the objects that match that charecteristc.*
   Unlike the previous one, many objects could have the same name, which means, that it will return in each element from the Array an object, for example:

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

   this will tell you how many objects have the name "usuario". For a simple example go into www.google.com and put:

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

We will see that the button called *btnG* now says "I rule!" , we changed its value.

3. **document.getElementsByTagName(TAG)[];**
Returns an *Array with the objects that match the characteristcs*.

This may be, the most useful one, it return objects witch TAG match the specified one, for example:

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

this returns the url of the first image in the document.

4. **By Groups:**

a.- document.frames

Returns an array with the frames that are in the document, only available in Internet Explorer.
It returns in order of appearance, the <iframe> y and the <frame>, if none, it returns null.

b.- document.forms

Returns an array with the forms that are in the document, it is important to clarify, that to obtain the object of an element of a form, we must access it through the form, for example:

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

Where usuario is the name of the form field.

c.- document.links

Returns all the links in the document, in 90% of the cases its the same as obtaining it using document.getElementsByTagName("a"); it is important to clarify, that in Internet Explorer you can use "click();" that simulates the click of the mouse, but firefox doesn't allow it.

d.- document.layers

Presented originally in Netscape, and then adapted to Internet Explorer, is obtaining the tags "<layer>"and " " its use is very weird, but it could be useful for sites that use these tags.

e.- document.applets

To obtain control of the java applets, and interact with them, you can obtain the object with document.applets which is identical to getElementsByTagName("applet");

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

   Microsoft Internet Explorer allows you to obtain objects in a simple way, only using the ID, but this method is NOT portable and I don't recommend it.

Obtaining PROPERTIES of OBJECTS:

Simply call it like this:
OBJETO.PROPIEDAD;

Some modifiable properties which are useful:

- **src**

  This is helpful to obtain the value of "src" or source of an image, movie, sound, script, etc.. especially when the "context menu" of the document is disabled. (Which we'll see how to re-enable later on), also you can modify it.

- **href**

  This will return, if it's a link, where does it point, useful in "document.links", also modifiable.

- **value**

  It returns the value of a form field, and we can modify it as we like, it's useful, for example, to discover the value of a hidden form field and to modify it like this:

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

- **length**

  In case of being an array or an object, it will return the amount of properties that it has, in case of being a string, it returns the length of the string.

- **maxLength**

  One of the most annoying protections when we're going to do an attack, without the need to modify the file or the headers, we only need to disable maxLength by giving it a null value, or giving it a very high value or giving it "-1".

- **readOnly**

  When they don't want to use hidden fields, the put it with the readOnly option or the disable the field, these could also be disabled, just give it a "false" value.

- **disabled**

  Just like readOnly, only need to give it a 0 value or a false, and you will be able to edit that form field.

- ***Actions inside the forms.***

  - **focus()**

    It helps you detect a field that you can't find, the focus() function will place cursor in the desired object.

  - **click()**

    Only for InternetExplorer, simulates a click, but in Firefox you can emulate an event, we will see this later on.

  - **select()**

    Functions similar to focus, but is usefull when working with formularie fields and menus, for example, a field type radio, will select when calling this function, and an option of a select field will take the place of it.

We can also assign that a set of commands will be from the properties or attributes of an specific object, if we put the object inside a with() , for example:

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

Will show us the title and URL of the document.

About the object document

This has several properties very interesting, which could be useful, like:

- **document.scripts**
  Returns the scripts in the document.
- **document.embeds**
  Returns the embed objects, like videos, flash objects, etc..
- **document.mimeType**
  Returns the mimeType of the document.. Usually is *HTML Document*.
- **document.protocol**
  Returns the document protocol.
- **document.URL**
  Returns the URL.. Also you can obtain the URL without the urlencode in *document.URLUnencoded*.
- **document.images**
  Returns the images in the document.
- **document.anchors**
  Returns the sections in the document.
- **document.selection**
  Returns what's selected in the document.
- **document.security**
  If it's HTTPS returns if it has a valid certificate.

Some of this properties may not be available in your browser.

document.location

Other properties that can be used, for control of the navigation:

- **document.location.href** – gives you the url of the document.
- **document.location.reload()** - reloads the site, same as the "refresh" button.
- **document.location.hostname** – returns the hostname (obviously).
- **document.location.host** – returns the host..
- **document.location.hash** – returns the section of the anchor of the location of the document, what's after the #.
- **document.location.search** – returns what's after ? and before #, in the url, the variables passed with GET.
- **document.location.port** – returns the port of the url.. but, it needs to be specified in the URL. For example: *protocolo://hostname.host.domain:puerto/path?var=val#anchor*
- **document.location.pathname** – returns the path of the file.

- **document.location.protocol** – returns the protocol section of the document, http: ftp: file: etc..
- **document.domain** – it returns on what domain the document is being executed, some functions don't have access if the domain of the documents destine is different from the original.
- **document.referrer** –returns the refferer of the document.
- **window.history**
  - **length**

    Returns the cuantity of pages saved in the history of the current sesion.

  - **go(int);**

    Takes you to the position indicated in the history .

  - **back(int);**

    It will take you back as many times as specified.

  - **forward(int);**

    It will take you forward the amount of times specified.

Note:
Frames have a document separate from the original, there for its value could change.

Obtaining all the properties of an object.

Sometimes you need to know the properties of an object.. for that you could use the next sintax:

```
for(var x in OBJECT){
   x=PROPERTY_OF_OBJECT;
}
```

example:

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

This function is similar to the foreach in other languages.

It's also worth mentioning that to access the properties of the object you access it like an array... there for, to obtain the properties value it would be like this:

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

Accesing OBJECTS by levels

You use the next attributes:

1.- ***self***
refers to the actual document
2.- ***top***
refers to the main document
3.- ***parent***
refers to the parent object

Replacing values.

If we want to modify values in the source code, or some property in a recurrent way, not doing it one by one...

The best way is using split() and join(), for example:

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

You can also use the replace(); function

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

This will change all the "a" for "A".
It's used generally to disable some scripts:

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

You can also use regular expressions in JavaScript:

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

The function replace(); also acepts regular expressions.

For more information on regular expressions visitr:

http://en.wikipedia.org/wiki/Regular_expression

Obtaining/Creating Styles:

There are 2 ways of doing this, with:
1.- ***currentStyle***
sintax:
OBJECT.currentStyle[PROPERTY];

2.- ***getComputedStyle***
sintax:

```
document.defaultView.getComputedStyle(OBJECT,null).getPropertyValue(PROPE
RTY);
```

We can generate style sheets with:

document.createStyleSheet();
and then access the css code with the cssText property.

Obtaining the size and position of an object:

1.- Using the previous mentioned functions of Style

- style.top
- style.left
- style.height
- style.width

2.- With the offset properties:

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

These are used commonly to obtain the size of the screen.

Obtain/Modify objects source code:

There are 2 almost identical ways of obtaining it:
1.- *OBJECT****.innerHTML;***
This obtains the inner HTML source code.
2.- *OBJECT****.outerHTML;***
This obtains the HTML code that is generated by the inner HTML.

The same way you can obtain the text without the HTML code:

1.- *OBJECT*.***innerText;***
Obtains the document's text.
2.- *OBJECT*.***outerText;***
Obtains the text generated by the inner HTML.

Add/Replace/Remove segments of the code.

With JavaScript we can modify only the desired sections of code, for example, remove divs, add them, replace them, clone them, etc...

In the next example, we'll add a new form field at the end of the document:

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

Here we used 3 functions, that I'll explain them:

document.createElement
This generates an HTML virtual object.

setAttribute
Adds an attribute to a virtual object.

appendChild
Adds to the virtual object, a real object, and generates it.

There are also other functions that can help you:

replaceChild(object1,object2);
Replaces object1 with object2.

removeChild(objeto);
Removes the object from the document.

cloneNode(bool);
Clones an attribute, and saves it in a variable, like a virtual object.

Obtain/Generate dynamic functions.

To obtain the code of a javscript function, even if it's encrypted, it could be obtained the following way:

alert(funcion);

The same technique could be used to obtain “classes”

Normally there are very anoying functions that erase the clipboard or any other thing that bothers us when we are examining a website.

These functions can be redeclared, for our facility.

You just have to redeclare them, for example, to disable right clicking, it’s common to use something like this:

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

To obtain the code of the function we do:

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

And to re habilitate it, we put:

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

or we can reassign the event, like we’ll see later on.

Another way of declaring variables:

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

or this way:

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

Asume control of events.

Asume events means that you take control over an event, this is handled differently in Netscape, using the document.captureEvents(); functions, but, we’ll be using Mozilla FireFox and Internet Explorer , that use the events like properties of an object...

For example, window.onunload is called when the window closes.

document.oncontextmenu is called when you right clic the document.

Now, this is a program that disables right clic, we'll see that with us asuming the document.oncontextmenu event we can evade this protection.

The code:

```
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");
```

This code is considerably safe, jumps some plugins that re-enable right clic.... but it can't beat JaSiLDBG.

All we have to do is:

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

We can also assume control of onunload, onbeforeunload, onmousedown, etc.... for example, there are codes that verify if the charecter you wrote in a field is valid, and if it's not, it doesn't capture it... even dough this looks like it's going to change in future versions of javascript, because of the danger it presents to users privacy.... but, these codes are disabled modifying the next events:

Onkeypress

Onkeydown

Onkeyup

Onchange

Like we already saw, to obtain an object:

Supose this is one unique form, and it's field is called "usuario"

Which means.. it would be:

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

To obtain all the events of a document, you can see the W3C standards or with the tools of JaSiLDBG, that you'll find at the end of this document.

Stop/Create Intervals.

Another technique used to "stop" you from copying code, or print, is clearing the clipboard.. the following code does this:

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

Here, an interval is generated, so that in every 100 miliseconds, the clipboard gets errased.

To clear the interval, we use the clearInterval(); function

Thus, for the code used in the example, the Interval gets cleared just by putting:

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

We can also create are own Intervals, so a command could be executed each period of time.

It's done like this:

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

To take control of an interval that wasn't saved into a variable, you can use this code: `javascript:void(clearInterval(setInterval("",0)-1));`
What it does, is take control of the last interval generated.
If you have been following this document from the beginning, you could also say, that you can disable the Interval from the example, overwriting the clrtmr() function, if you thought about it, we're doing good ☺.

Obtain source of images, scripts, movies, etc..

To obtain the url of an image, movie, etc.. you could:

Simply use:
*OBJECT.**src;***

But with frames you use:
*OBJECT.**location;***

And for links, and other (if you want to obtain the anchor you can use OBJECT.**hash**):

*OBJECT.**href***

Inclusive, with frames, or any other document, you could obtain the referrer:
*OBJECT.**document.referrer;***

For example, to obtain the source of an image, you could use:

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

it will obtain the source of the first image in the document.

Obtain complete document as an object.

There are 2 ways, use whitch ever you please:
1.- *document.getElementsByTagName("html")[0];*
2.- *document.documentElement;*

This could be useful to obtain the real source code of the document, the following way:

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

Working with Shockwave Flash objects

Flash, can interact with javascript and viceversa, even dough the comunication javascript=>flash is more complicated than actionscript=>javascript, it's possible. Some of the functions that control ActiveX of Flash and LCn of Mozilla are:

Control of a movie:

1. ***IsPlaying();***
   State of the movie.
2. ***Play();***
   Starts the movie.
3. ***Stop();***
   Stops the movie.
4. ***Back();***
   Goes back a frame in the movie.
5. ***Forward();***
   Avances one frame in the movie.
6. ***Rewind();***
   Returns the movie to the first frame.
7. ***GotoFrame(int);***
   Makes the movie go to the specified frame.
8. ***FrameLoaded(int);***
   Returns true if the movie is in the specified frame.
9. ***CurrentFrame();***
   Returns the current frame.

Control of the producer:

1. ***FlashVersion();***
   Returns the version of the flash playing.
2. ***LoadMovie(int,URL);***
   Loads a certain movie (int) of the specified *URL*.
3. ***PercentLoaded();***
   Returns the percent loaded of the movie.
4. ***SetZoomRect(LEFT,UP,RIGHT,DOWN);***
   Establishes the zoom in the specified zone.

Control of program and properties:

1. ***SetVariable(VARIABLE,VALUE);***
   Introduces in the *VARIABLE* the specified *VALUE*.

2. ***GetVariable(VARIABLE);***
   Returns the value of *VARIABLE*.
3. ***TCurrentFrame(INST);***
   Returns the frame of certain instance.
4. ***TCurrentLabel(INST);***
   Returns the name of the frame.
5. ***TCallFrame(INST,int);***
   Calls the function in this frame.
6. ***TCallFrame(INST,NAME);***
   Calls the function in the frame with that NAME.
7. ***TGetProperty(INST,PROPERTY);***
   Returns the instante property.
8. ***TSetProperty(INST,PROPERTY,VALUE);***
   Establishes certain *VALUE* to the *PROPERTY* of the instance.

Access an object/variable

If the object identifier is “road”, you can access the object (or the variable) the following way:

/:road

so for changing the position of the object, the code would be like this:

```
      javascript:void(document.getElementsByTagName("embed")[0].TSetPrope
rty("/:road",0,100));
```

To change a property/variable of the object, would be like this:

```
      javascript:void(document.getElementsByTagName("embed")[0].SetVariab
le("/:road:variableorpropertyname","value"));
```

Discover unknown variables

The demo version of the program “flash-decompiler” of Eltima software, allows you to see inside the SWF, and its Action Script code, you can free download it, from: http://www.flash-decompiler.com/.

Check references.

Other methods of this handler can be found at the flash support center.

A PlayGround for JaSiLDBG and Flash, can be found at the excercices and examples for JaSiLDBG website.

Working with ActiveX controlers

The Microsoft ActiveX controlers can interact with javascript, the OCX object that you create must have its functions and properties in “Public” mode, and are called the fallowing way:

OBJECT.***function();***
OBJECT.***property;***

OBJECT must point to an <OBJECT> that controls the ActiveX object, it is important to clarify that many bugs in the ActiveX controls exist because the programmer thinks, that its imposible to call a function on the same domain without breaking signatures of the integrity of the code. This is obviously false with JaSiLDBG, through JaSiLDBG, we can open an MSN Messenger conversation window from the Hotmail page, we can start games in the same, also with sites like "www.gamedesire.com" we have the capability of modifying properties that suposed to be secure, also values in casinos sites, that save in an ActiveX control essential values (for more security), that modifying them you can compromise the Casino, and that they think that by using SSL they are free from this (same thing with some banks). Sadly, it isn’t this way, at least with JaSiLDBG the capabilities are develoble in the ActiveX controler area, but since there are so many controlers its imposible mentionaning all of them, and its very easy to exploit, once you understood the concept.

Modifying and obtaining Headers with javascript & XMLHTTP.

Depending on your explorer, the objects that namage XMLHTTP are different.

For Opera and Firefox:

var cx = new *XMLHttpRequest();*

IExplorer:

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

Now that we have the object, we can, for example, make a petition to the current website.

(YOU CAN ONLY DO PETITIONS OF AJAX TO THE SAME DOMAIN WHERE YOU ARE)

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

This is:

1. Establish that "cx" will have the XMLHTTP object.
2. We open with the "*GET* " method the current directory.
   the false is so that it will wait until the document is ready.
3. We send the petition.
4. And we show the response we got.

There are more methods, besides GET, for example:

- **POST,** and the POST information you send it with the next function:
  send("INFO=AQUI&MAS=MAS");
  it's important to clarify that for it to work, and be recognized by the server, you must add the header, "Content-Type" with "application/x-www-form-urlencoded".
- **HEAD**, only returns the header.
- **SEARCH**, not available in all.
- **TRACE**, disabled in the latest XMLHTTP.. returned the petition that we sended the server, the use of this header, with attacks of XSS (Cross Site Scripting), take place in another type of attack, XST (Cross Site Tracing).

And anything else your server might support.. must be uppercase.
to send a personalizad header you could do it like this:
cx.setRequestHeader("Referrer: http://www.hey.com/");
after *open()* and before *send();*
to retrieve all the headers that the Server responded:
cx.getAllResponseHeader();
after *send();*
to receive only one for example:
cx.getResponseHeader("Content-type");
after *send();*

Its also important to clarify, you can execute commands one by one.. it doesn't have to be all at one.. for example:

In the current site put:

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

then:

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

and then you can use the cx object without needing to re-load it each time.

Other uses of JaSiLDBG.

In our experience, we have found, that JaSiLDBG can be useful for many things, more than what we have been talking about.. but, given that capabilities are ilimted, we only mention some cases where it was useful.

Calculator

Simply using the Math library, or using simple functions, like adding and subtracting, multiplying, etc.. or even obtain modular computations, and binary, obtain values of constants, such as PI or E, create random numbers, round up, etc.. you can find a list of all the Math functions in this link:

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

Cryptoanalysis

Lately, we've used JaSiLDBG for cryptografy, for example, Crack_X made a tool called Pescadito, that encrypts selected text in differnt algorithms, such as blowfish twofish, AES, or creates checksums like MD5 o SHA-1, and Sirdarckcat, thanks to JaSiLDBG, was able to break with success the algorithm of an encryption used in a website that has a points system.

A very helpful function, for this topic:

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

charCodeAt(int);

returns there ASCII value of the charecter selected.

We also have other mathematical operations, such as XOR, that is interpreted by the symbol ^, AND whos symbol is &, OR whitch is |, or for modular criptograhpy, %.

Removing Publicity

Many times, pulicity, deceptive, or just anoying , dont allow us to fully view a website, or obligate us to download an ActiveX controller that is very dangerous, and that without it, we cant download the desired file, or sites that when they close, they re-open.. this is very common, and assuming events, like "onUnload" or "onBeforeUnload" we could stop these bucles.. we can also hide divs using:

Object.style.visibility=”Hidden”;

Debugging navigators

During the elaboration of this document, 2 bugs where discovered in Microsoft Internet Explorer, of NULL pointer exception, that could end in Buffer Overflow, and with that, remote code executation.

Create Regular Expressions

JavaScript, has a function called RegExp, that permits creating regular expressions, document.createExpression also exists, you can find an example and how to use it in the following link:

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

Modify websites with one clic

Using comands of JaSiLDBG from a bookmark, we can modify a websites aspect, with only one clic. Wikipedia, for example, to add more interaction when creating articles, has promoted libraries to add options.. Also throught time commands have been used in bookmarks to evade the payment of animated Meegos from Microsoft © or to jump the genuine software verification of Windows©, among other things.

Automate Actions

Simply with a macro, to generate loops, from a simple flood, to generating brute-force attacks.. or filter some data.

Exploiting XSS, SQLi, RFI, etc..

Left for last, but is the most important, the use of hidden values, or cookies, that we already saw that can be modified, contains information, that they believe that having sessions, control over referrers, can’t be faked, but with JaSiLDBG, it’s not true..

JaSiLDBG Library, estigma.js

We've developed a small tool, with many functions that could help to, investigate headers that the Server responds, edite a document like it was frontpage or dreamwaver, use some cryptography algorithms, like AES, Blowfish, you can also do md5 checksums, or convert password fields, or hidden fields to text.. etc..

Just write this script in your address bar:

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

I recommend to add it as a bookmark.

The list of functions that it brings:

resHeaders()
Returns the headers that are responded with a Get Standard petition, to the current document.

fleHeaders(*file*)
Returns the headers that are responded with a Get Standard petition, to the current file.

resSource()
Returns the source code of the current document.

fleSource()
Returns the source code of the specified document.

emReq()
Returns an XMLHTTP handle, to the current document.

explain()
Returns a string with the properties and its values of an object.

moCookies()
Returns a series of dialogs to modify cookies individually.

geEvents()
Returns a string with the events that an object has.

geGET()
Returns an array with the variables sent in a Get petition.

geCookies()
Returns an array with the variables in the cookie.

page(*obj*,*num*,*sort*)
Returns a string, that orders the elements of an array.

xpage(*obj*)
Returns, a series of dialogs with the properties of an array.

LoadLib(*url*)
Loads the functions of the library specified.

Pescadito()
Loads the criptographyc functions of Pescadito, by Crack_X

disMax()
Disables the max charecter limits in all the form fields.

disPass()
Converts all the password form fields to text fields.

disHide()
Makes visible all the hidden form fields.

disFreeze()
Enables all the form fields with property readOnly or disabled.

disAll()
Executes, the functions disMax(), disPass(), disHide(), disFreeze().

geCSS()
Returns the style sheet that's in the document.

geScripts()
Returns the scripts in the document.

EditHTML()
Loads at the end of a document, an iframe with WYSIWYG, to edit the body of the document.

ApplyHTML()
After doing all the changes with the EditHTML() function, this function will apply the changes.

ASCII()
Returns the ASCII value of every character in a string.

dec2hex()
Converts a decimal to hexadecimal.

dec2bin()
Converts a decimal to binary.

hex2dec()
Converts an hexadecimal value to it's decimal value.

hex2bin()
Converts a hexadecimal value to it's binary equivalent.

bin2dec()
Converts a binary value to decimal.

bin2hex()
Converts a binary value to hexadecimal.

Other functions may have been added, to see the last function list, enter to:

http://sirdarckcat.googlepages.com/estigma

References

JavaScript References

- Mozilla
- DevGuru

DOM Inspector

- Mozilla

HTML References

- W3c

Shockwave Flash References

- Scripting with flash

ActiveX Controlers

- ActiveX Objects

Excersice examples for JaSiLDBG

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

Javascript Protocol

- Microsoft Developer Network (MSDN)

Discusion about this document

- at *foro.elhacker.net*

About this document

The document was developed through Zoho Writer, then exported to Word, and published in Google Docs.

For the final version, a PDF was generated for the English and for the Spanish version.

The last version of JaSiLDBG can be found at:
http://jasildbg.googlepages.com/en

Authors

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/

License

Thanks

We want to thank the STAFF from elhacker.net forum for giving us suggestions, corrections and support; mainly Alex Bove.