{

"name": "CounterXSS",

"version": "1.1",

"description": "An Extension to counter XSS attack!",

"manifest\_version": 2,

"browser\_action": {

"default\_icon":"icon.png",

"default\_popup":"popup.html"

}

}

<!DOCTYPE html>

<html>

<head>

<title>CounterXSS</title>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<link rel="stylesheet" type="text/css" href="mystyle.css">

<style>

body {

background-color: linen;

text-align: center;

}

h1 {

color: maroon;

margin-left: 40px;

}

</style>

</head>

<body>

<h1>Background scanning in progress</h1>

<div>

<button type="button" class="btn btn-danger" id="button" onclick="securityhttp()">Scan Again</button>

</div>

</body>

</html>

var myString = document.getElementById("str").innerHTML;

var myWord = document.getElementById("txt1").value;

var myPattern = new RegExp('(\\w\*'+myWord+'\\w\*)','gi');

var matches = myString.match(myPattern);

if (matches === null)

{

document.getElementById("demo").innerHTML = "No results"; // Any message or empty

return;

}

document.getElementById("demo").innerHTML = matches + " - " + matches.length + " result(s) found.";

function searchScript (str, strArray) {

for (var j=0; j<strArray.length; j++) {

if (strArray[j].match(str)) return j;

}

return -1;

}

function securityhttp() {

if (location.protocol != 'https:')

{

location.href = 'https:' + window.location.href.substring(window.location.protocol.length);

}

else

alert("Insecured website");

}

function Hilitor(id, tag)

{

var targetNode = document.getElementById(id) || document.body;

var hiliteTag = tag || "EM";

var skipTags = new RegExp("^(?:" + hiliteTag + "|SCRIPT|FORM|SPAN)$");

var colors = ["#ff6", "#a0ffff", "#9f9", "#f99", "#f6f"];

var wordColor = [];

var colorIdx = 0;

var matchRegex = "";

var openLeft = false;

var openRight = false;

// characters to strip from start and end of the input string

var endCharRegex = new RegExp("^[^\\\w]+|[^\\\w]+$", "g");

// characters used to break up the input string into words

var breakCharRegex = new RegExp("[^\\\w'-]+", "g");

this.setMatchType = function(type)

{

switch(type)

{

case "left":

this.openLeft = false;

this.openRight = true;

break;

case "right":

this.openLeft = true;

this.openRight = false;

break;

case "open":

this.openLeft = this.openRight = true;

break;

default:

this.openLeft = this.openRight = false;

}

};

this.setRegex = function(input)

{

input = input.replace(endCharRegex, "");

input = input.replace(breakCharRegex, "|");

input = input.replace(/^\||\|$/g, "");

if(input) {

var re = "(" + input + ")";

if(!this.openLeft) re = "\\b" + re;

if(!this.openRight) re = re + "\\b";

matchRegex = new RegExp(re, "i");

return true;

}

return false;

};

this.getRegex = function()

{

var retval = matchRegex.toString();

retval = retval.replace(/(^\/(\\b)?|\(|\)|(\\b)?\/i$)/g, "");

retval = retval.replace(/\|/g, " ");

return retval;

};

// recursively apply word highlighting

this.hiliteWords = function(node)

{

if(node === undefined || !node) return;

if(!matchRegex) return;

if(skipTags.test(node.nodeName)) return;

if(node.hasChildNodes()) {

for(var i=0; i < node.childNodes.length; i++)

this.hiliteWords(node.childNodes[i]);

}

if(node.nodeType == 3) { // NODE\_TEXT

if((nv = node.nodeValue) && (regs = matchRegex.exec(nv))) {

if(!wordColor[regs[0].toLowerCase()]) {

wordColor[regs[0].toLowerCase()] = colors[colorIdx++ % colors.length];

}

var match = document.createElement(hiliteTag);

match.appendChild(document.createTextNode(regs[0]));

match.style.backgroundColor = wordColor[regs[0].toLowerCase()];

match.style.fontStyle = "inherit";

match.style.color = "#000";

var after = node.splitText(regs.index);

after.nodeValue = after.nodeValue.substring(regs[0].length);

node.parentNode.insertBefore(match, after);

}

};

};

// remove highlighting

this.remove = function()

{

var arr = document.getElementsByTagName(hiliteTag);

while(arr.length && (el = arr[0])) {

var parent = el.parentNode;

parent.replaceChild(el.firstChild, el);

parent.normalize();

}

};

// start highlighting at target node

this.apply = function(input)

{

this.remove();

if(input === undefined || !input) return;

if(this.setRegex(input)) {

this.hiliteWords(targetNode);

}

};

}