

XCTF-攻防世界CTF平台-Crypto类——4、 flag_in_your_hand1（前端输入加密）

原创

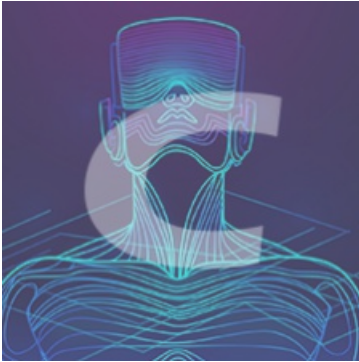
太...白 于 2021-09-12 14:22:34 发布 153 收藏 2

分类专栏: [# Bugku、XCTF-Crypto类CTF写题过程](#) [# Bugku、XCTF-WEB类写题过程](#) 文章标签: [html](#) [javascript](#) [html5](#) [算法](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/Onlyone_1314/article/details/120250146

版权



[Bugku、XCTF-Crypto类CTF写题过程](#) 同时被 2 个专栏收录

2 篇文章 0 订阅

订阅专栏



[Bugku、XCTF-WEB类写题过程](#)

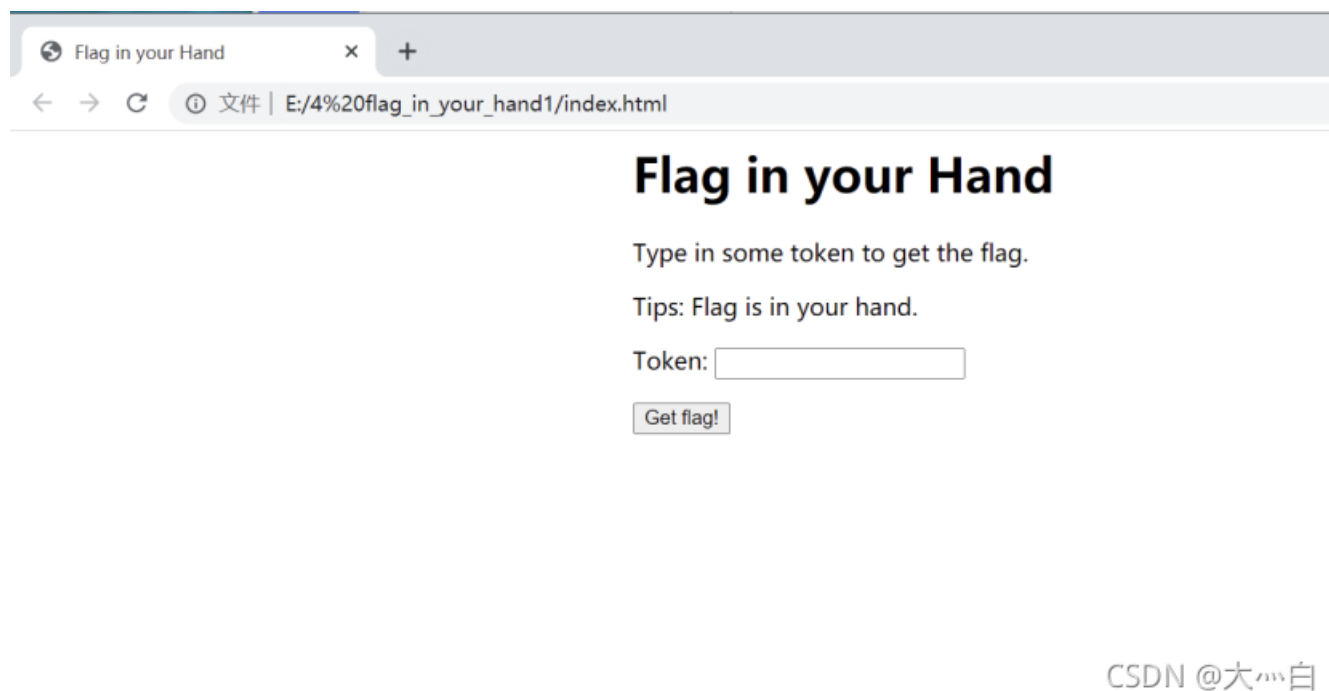
24 篇文章 2 订阅

订阅专栏

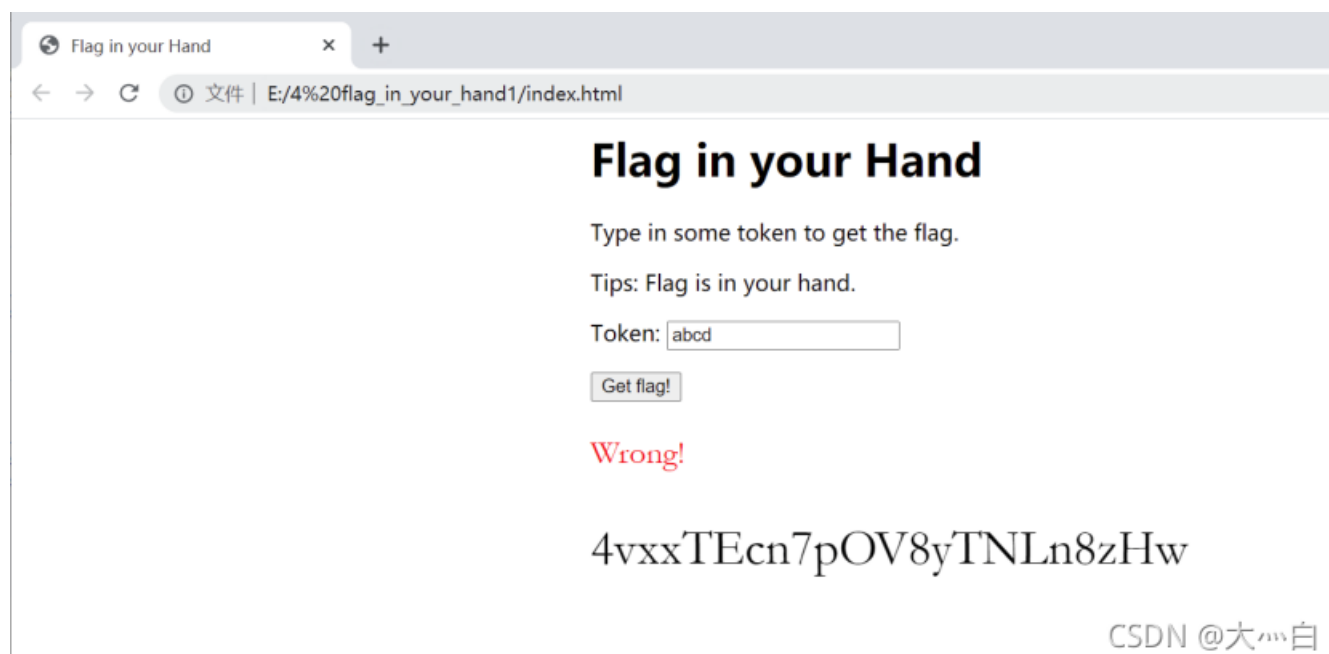
下载文件之后, 发现是一道前端的题目: 有一个index.html文件和一个script-min.js文件:

index.html	2021/9/2 22:57	SLBrowser HTML D...	2 KB
script-min.js	2018/8/27 14:29	JavaScript 文件	8 KB

运行html文件:



要求输入一个Token, 然后点Get flag!按钮:



代码逻辑就是输入一个字符串，然后经过计算对比得到正确的flag

我们先查看源代码：

```
index.html x script-min.js x
26 </style>
27 <script src="script-min.js"></script>
28 <script type="text/javascript">
29     var ic = false;
30     var fg = "";
31
32     function getFlag() {
33         var token = document.getElementById("secToken").value;
34         ic = checkToken(token);
35         fg = bm(token);
36         showFlag()
37     }
38
39     function showFlag() {
40         var t = document.getElementById("flagTitle");
41         var f = document.getElementById("flag");
42         //!!判断ic是否为空，为空则t.innerText赋为第一个值
43         t.innerText = !!ic ? "You got the flag below!!" : "Wrong!";
44         t.className = !!ic ? "rightflag" : "wrongflag";
45         f.innerText = fg;
46     }
47 </script>
48 </head>
49 <body>
50     <h1>Flag in your Hand</h1>
51     <p>Type in some token to get the flag.</p>
52     <p>Tips: Flag is in your hand.</p>
53     <div>
54         <p>
55             <span>Token:</span>
56             <span><input type="text" id="secToken"/></span>
57         </p>
58         <p>
59             <input type="button" value="Get flag!" onclick="getFlag()" />
60         </p>
61     </div>
62     <div>
63         <p id="flagTitle"></p>
64         <p id="flag"></p>
65     </div>
66 </body>
67 </html>
```

Hyper Text Markup length: 1,422 lines: 67 Ln: 59 Col: 74 Sel: 9 | 1 Windows (CR LF) UTF-8

```
<html>
<head>
<title>Flag in your Hand</title>
<style type="text/css">
body {
padding-left: 30%;
}

#flag {
font-family: Garamond, serif;
font-size: 36px;
}

#flagtitle {
font-family: Garamond, serif;
font-size: 24px;
}

.rightflag {
color: green;
}
```

```

.wrongflag {
  color: red;
}
</style>
<script src="script-min.js"></script>
<script type="text/javascript">
  var ic = false;
  var fg = "";

  function getFlag() {
    var token = document.getElementById("secToken").value;
    ic = checkToken(token);
    fg = bm(token);
    showFlag()
  }

  function showFlag() {
    var t = document.getElementById("flagTitle");
    var f = document.getElementById("flag");
    ///!!判断ic是否为空, 为空则t.innerHTML赋为第一个值
    t.innerHTML = !!ic ? "You got the flag below!!" : "Wrong!";
    t.className = !!ic ? "rightflag" : "wrongflag";
    f.innerHTML = fg;
  }
</script>
</head>
<body>
  <h1>Flag in your Hand</h1>
  <p>Type in some token to get the flag.</p>
  <p>Tips: Flag is in your hand.</p>
  <div>
    <p>
      <span>Token:</span>
      <span><input type="text" id="secToken"/></span>
    </p>
    <p>
      <input type="button" value="Get flag!" onclick="getFlag()" />
    </p>
  </div>
  <div>
    <p id="flagTitle"></p>
    <p id="flag"></p>
  </div>
</body>
</html>

```

输入框的id="secToken"，值保存在token变量中，先调用checkToken(token)函数检查token
查看checkToken()函数：

```

function checkToken(s) {
  return s === "FAKE-TOKEN";
}

```

它在script-min.js只有这一处声明，是个假的TOKEN返回给ic变量，下面showFlag()函数也是根据ic变量的值判断输出flag的之后的fg = bm(token);就是根据输入的字符串计算得到一个最终的结果，之后在前端最下面显示出来注意这里的ic变量的值和fg变量的值是没有关系的，但是最终输出flag的时候他们都要求是正确的值先求flag字符串内容的过程：fg变量查看bm(token)函数：

```
1 function hm(s) {
2     return rh(rstr(str2rstr_utf8(s)));
3 }
4 function bm(s) {
5     return rb(rstr(str2rstr_utf8(s)));
6 }
7 function rstr(s) {
8     return binl2rstr(binl(rstr2binl(s), s.length * 8));
9 }
10 function checkToken(s) {
11     return s === "FAKE-TOKEN";
12 }
13 function rh(ip) {
14     try {
15         hc
16     } catch (e) {
17         hc = 0;
18     }
19     var ht = hc ? "0123456789ABCDEF" : "0123456789abcdef";
20     var op = "";
21     var x;
22     for (var i = 0; i < ip.length; i++) {
23         x = ip.charCodeAt(i);
24         op += ht.charAt((x >>> 4) & 0x0F) + ht.charAt(x & 0x0F);
25     }
26     return op;
27 }
28 function rb(ip) {
```

CSDN @大...白

它层层嵌套，调用了3个函数str2rstr_utf8(s)、rstr()、rb()来处理输入的字符串先看str2rstr_utf8(s)函数：

```
function str2rstr_utf8(input) {
    var output = "";
    var i = -1;
    var x, y;
    while (++i < input.length) {
        x = input.charCodeAt(i);
        y = i + 1 < input.length ? input.charCodeAt(i + 1) : 0;
        if (0xD800 <= x && x <= 0xDBFF && 0xDC00 <= y && y <= 0xDFFF) {
            x = 0x10000 + ((x & 0x03FF) << 10) + (y & 0x03FF);
            i++;
        }
        if (x <= 0x7F)
            output += String.fromCharCode(x);
        else if (x <= 0x7FF)
            output += String.fromCharCode(0xC0 | ((x >>> 6) & 0x1F), 0x80 | (x & 0x3F));
        else if (x <= 0xFFFF)
            output += String.fromCharCode(0xE0 | ((x >>> 12) & 0x0F), 0x80 | ((x >>> 6) & 0x3F), 0x80 | (x & 0x3F));
        else if (x <= 0x1FFFFFF)
            output += String.fromCharCode(0xF0 | ((x >>> 18) & 0x07), 0x80 | ((x >>> 12) & 0x3F), 0x80 | ((x >>> 6) & 0x3F), 0x80 | (x & 0x3F));
    }
    return output;
}
```

CSDN @大...白

这个函数只是将我们输入的字符串转换成utf-8编码格式

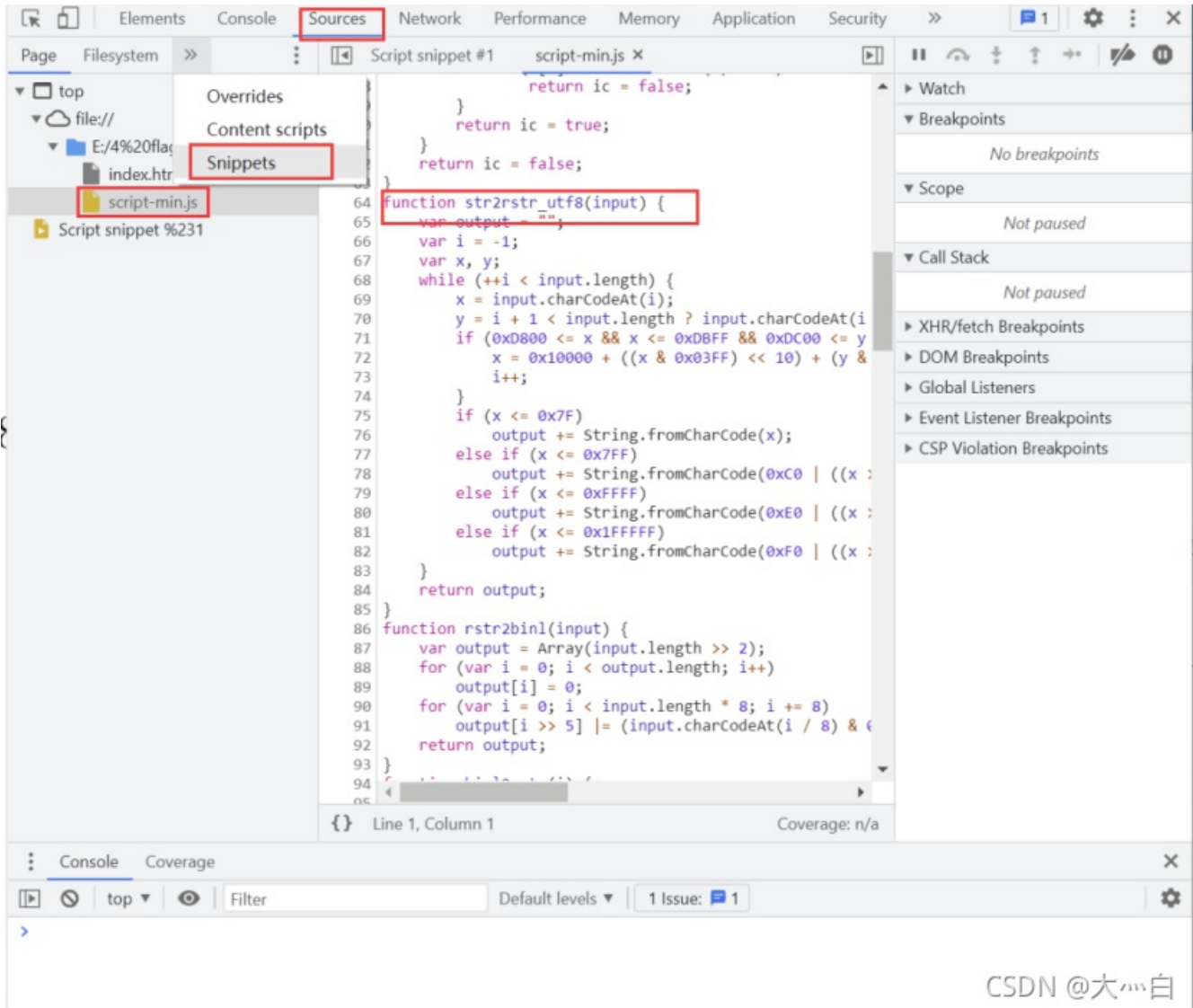
直接在控制台输入str2rstr_utf8(s)函数的代码之后

str2rstr_utf8("123");调用函数对字符"123"处理之后输入的结果是"123"

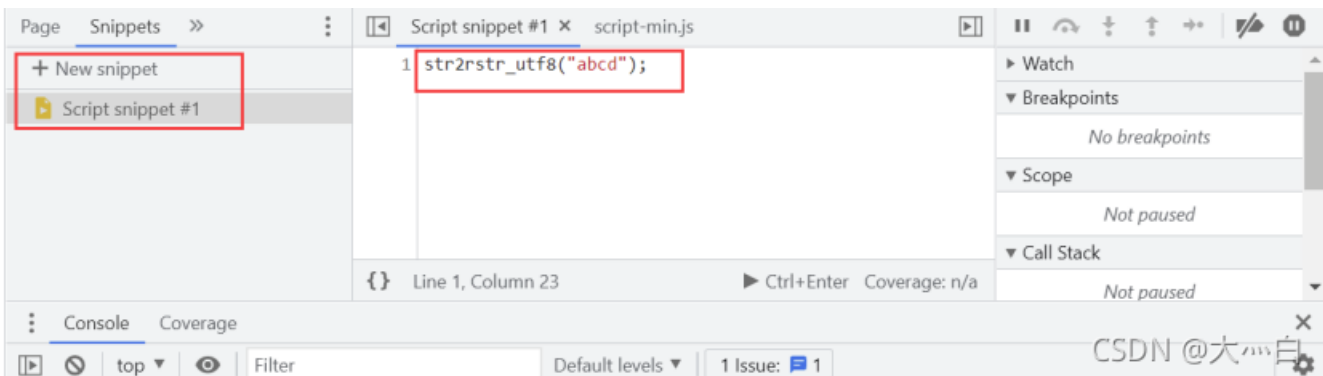
```
> function str2rstr_utf8(input) {
  var output = "";
  var i = -1;
  var x, y;
  while (++i < input.length) {
    x = input.charCodeAt(i);
    y = i + 1 < input.length ? input.charCodeAt(i + 1) : 0;
    if (0x0800 <= x && x <= 0xDBFF && 0xDC00 <= y && y <= 0xDFFF) {
      x = 0x10000 + ((x & 0x03FF) << 10) + (y & 0x03FF);
      i++;
    }
    if (x <= 0x7F)
      output += String.fromCharCode(x);
    else if (x <= 0x7FF)
      output += String.fromCharCode(0xC0 | ((x >>> 6) & 0x1F), 0x80 | (x & 0x3F));
    else if (x <= 0xFFFF)
      output += String.fromCharCode(0xE0 | ((x >>> 12) & 0x0F), 0x80 | ((x >>> 6) & 0x3F), 0x80 | (x & 0x3F));
    else if (x <= 0x1FFFFF)
      output += String.fromCharCode(0xF0 | ((x >>> 18) & 0x07), 0x80 | ((x >>> 12) & 0x3F), 0x80 | ((x >>> 6) &
0x3F), 0x80 | (x & 0x3F));
  }
  return output;
}
< undefined
> str2rstr_utf8("123")
< "123"
> str2rstr_utf8("123");
< "123"
< "123"
< "123"
< "abcd"
>
```

VM356 Script snippet %231:1
VM357 Script snippet %231:1
Script snippet %231:1
CSDN @大白

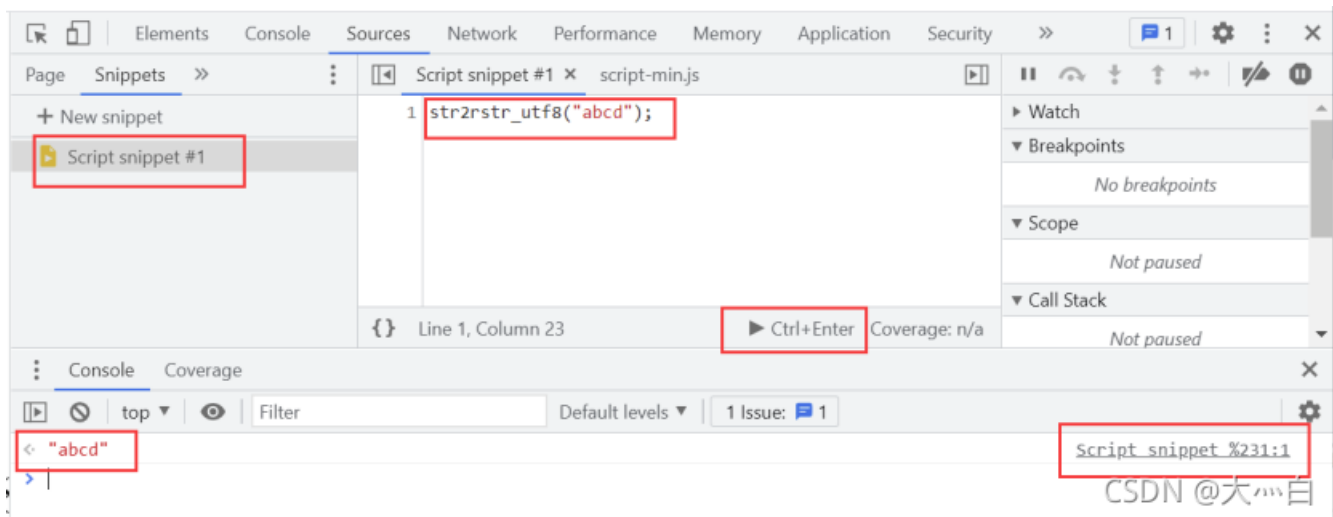
也可以直接在chrome浏览器里面新增js代码来调用原有的str2rstr_utf8(s)函数
在chrome浏览器的Sources->Snippets中



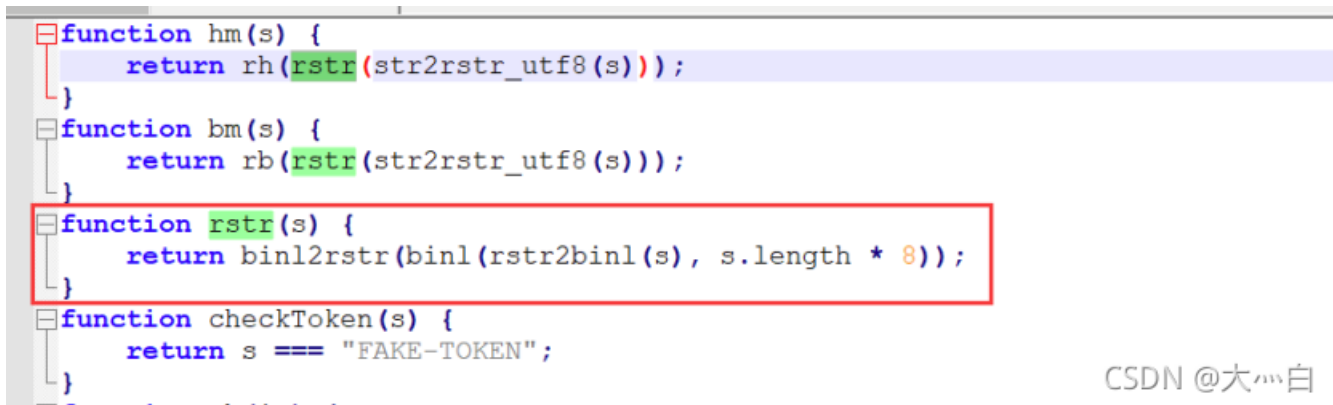
选择New snippet就能新建一个js文件
输入str2rstr_utf8("abcd");调用str2rstr_utf8(s)函数



之后点击Ctrl+Enter就能运行js代码:



之后就是rstr()函数:



CSDN @大白

依次调用了rstr2binl(s)、binl()、binl2rstr()函数
同样调用三个函数查看输出：

The screenshot shows a code editor with the following JavaScript code:

```
1 |  
2 //rstr2binl(str2rstr_utf8("abcd"));  
3  
4 //binl(rstr2binl(str2rstr_utf8("abcd")),4*8);  
5  
6 binl2rstr(binl(rstr2binl(str2rstr_utf8("abcd")),4*8));
```

The console output shows the following results:

```
< ▶ [1684234849] VM419 Sc  
< ▶ (4) [1282538722, -1813108921, -853216363, 523468590] VM421 Sc  
< "âüqLG'i@ó$í. 3\u001f"  
>
```

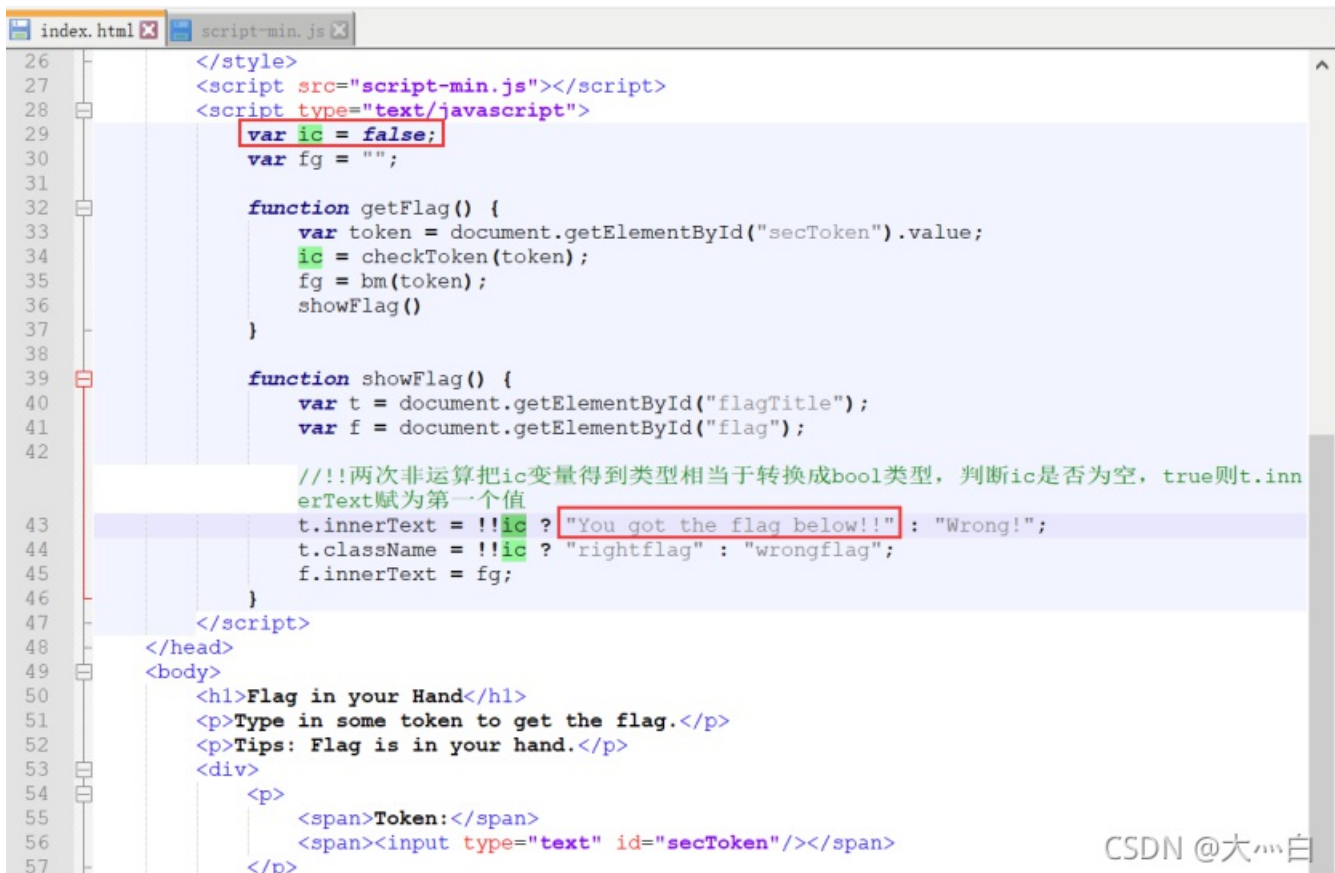
The console also shows a watermark: CSDN @大...白

发现他们对输入字符串做了复杂的处理，
查看rstr2binl(s)函数源代码：

```
function rstr2binl(input) {  
    var output = Array(input.length >> 2);  
    for (var i = 0; i < output.length; i++)  
        output[i] = 0;  
    for (var i = 0; i < input.length * 8; i += 8)  
        output[i >> 5] |= (input.charCodeAt(i / 8) & 0xFF) << (i % 32);  
    return output;  
}
```

```
function rstr2binl(input) {  
    var output = Array(input.length >> 2);  
    for (var i = 0; i < output.length; i++)  
        output[i] = 0;  
    for (var i = 0; i < input.length * 8; i += 8)  
        output[i >> 5] |= (input.charCodeAt(i / 8) & 0xFF) << (i % 32);  
    return output;  
}
```

从代码我们可以看到根据输入的字符串，决定了输出的字符串，所以现在无法直接逆向这部分代码。
接下来只能从ic变量去找突破口了，因为当我们输入的字符串是正确的字符串的时候，ic变量的值会被赋为true



```
26 </style>  
27 <script src="script-min.js"></script>  
28 <script type="text/javascript">  
29     var ic = false;  
30     var fg = "";  
31  
32     function getFlag() {  
33         var token = document.getElementById("secToken").value;  
34         ic = checkToken(token);  
35         fg = bm(token);  
36         showFlag()  
37     }  
38  
39     function showFlag() {  
40         var t = document.getElementById("flagTitle");  
41         var f = document.getElementById("flag");  
42  
43         //!!两次非运算把ic变量得到类型相当于转换成bool类型，判断ic是否为空，true则t.inn  
44         t.innerHTML = !!ic ? "You got the flag below!!" : "Wrong!";  
45         t.className = !!ic ? "rightflag" : "wrongflag";  
46         f.innerHTML = fg;  
47     }  
48 </script>  
49 </head>  
50 <body>  
51 <h1>Flag in your Hand</h1>  
52 <p>Type in some token to get the flag.</p>  
53 <p>Tips: Flag is in your hand.</p>  
54 <div>  
55 <p>  
56     <span>Token:</span>  
57     <span><input type="text" id="secToken"/></span>  
58 </p>  
59 </div>  
60 </body>  
61 </html>
```

可以看到ic变量的初始值被设为了false，但最后输出正确的flag的时候，要求它的值是true，所以我们去看ic变量的值在哪被更改了：

```
function ck(s) {
  try {
    ic
  } catch (e) {
    return;
  }
  var a = [118, 104, 102, 120, 117, 108, 119, 124, 48, 123, 101, 120];
  if (s.length == a.length) {
    for (i = 0; i < s.length; i++) {
      if (a[i] - s.charCodeAt(i) != 3)
        return ic = false;
    }
    return ic = true;
  }
  return ic = false;
}
```

CSDN @大...白

只有在ck(s)函数中有4次更改，也就是要输出正确的flag，ic的值就要在这被修改成true。实际上ck(s)函数是被binl(x, len)调用的：

```
function binl(x, len) {
  s = binl2rstr(x);
  x[len >> 5] |= 0x80 << ((len) % 32);
  x[(((len + 64) >>> 9) << 4) + 14] = len;
  var a = 1732584193;
  var b = -271733879;
  var c = -1732584194;
  var d = 271733878;
  for (var i = 0; i < x.length; i += 16) {
    var olda = a;
    var oldb = b;
    var oldc = c;
    var oldd = d;
    a = ff(a, b, c, d, x[i + 0], 7, -680876936);
    d = ff(d, a, b, c, x[i + 1], 12, -389564586);
    c = ff(c, d, a, b, x[i + 2], 17, 606105819);
    b = ff(b, c, d, a, x[i + 3], 22, -1044525330);
    a = ff(a, b, c, d, x[i + 4], 7, -176418897);
    d = ff(d, a, b, c, x[i + 5], 12, 1200080426);
    c = ff(c, d, a, b, x[i + 6], 17, -1473231341);
    b = ff(b, c, d, a, x[i + 7], 22, -45705983);
    a = ff(a, b, c, d, x[i + 8], 7, 1770035416);
    d = ff(d, a, b, c, x[i + 9], 12, -1958414417);
    c = ff(c, d, a, b, x[i + 10], 17, -42063);
    b = ff(b, c, d, a, x[i + 11], 22, -1990404162);
    a = ff(a, b, c, d, x[i + 12], 7, 1804603682);
    d = ff(d, a, b, c, x[i + 13], 12, -40341101);
    c = ff(c, d, a, b, x[i + 14], 17, -1502002290);
    b = ff(b, c, d, a, x[i + 15], 22, 1236535329);
    ck(s);
    a = gg(a, b, c, d, x[i + 1], 5, -165796510);
    d = gg(d, a, b, c, x[i + 6], 9, -1069501632);
    c = gg(c, d, a, b, x[i + 11], 14, 643717713);
    b = gg(b, c, d, a, x[i + 0], 20, -373897302);
    a = gg(a, b, c, d, x[i + 5], 5, -701558691);
    d = gg(d, a, b, c, x[i + 10], 9, 38016083);
  }
}
```

CSDN @大...白

也就是上面的计算flag的值的的过程中，调用了ck(s)函数去验证输入的字符串s，如果符合要求，就把ic的值设为true，然后在输出计算得到的flag。

我们接着分析ck(s)函数：

```
function ck(s) {
  try {
    ic
  } catch (e) {
    return;
  }
  var a = [118, 104, 102, 120, 117, 108, 119, 124, 48,123,101,120];
  if (s.length == a.length) {
    for (i = 0; i < s.length; i++) {
      if (a[i] - s.charCodeAt(i) != 3)
        return ic = false;
    }
    return ic = true;
  }
  return ic = false;
}
```

就是把输入的字符串的字符的ASCII码和a[]数组中的数逐个比较，一旦出现不是相差3的字符就返回

return ic = false;

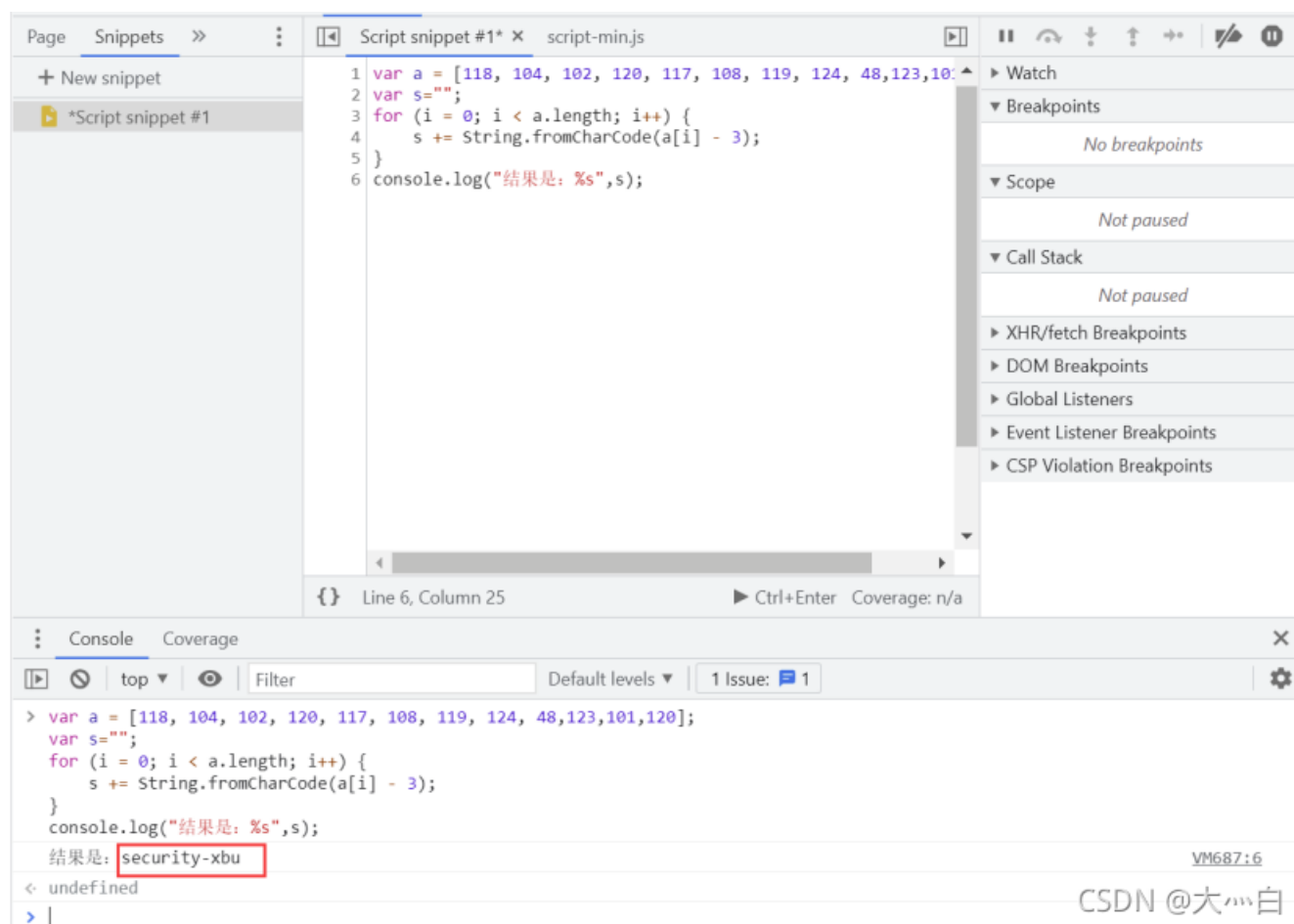
所有的字符和数字都相差3时就返回

return ic = true;

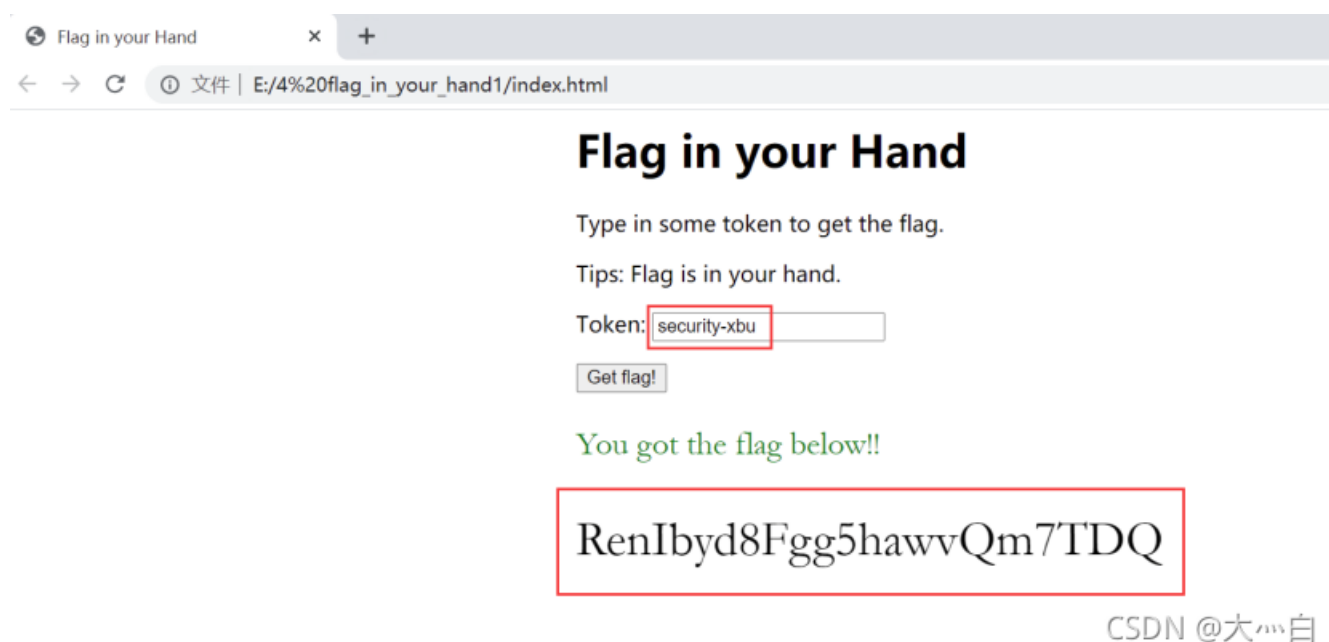
我们根据代码逻辑，写出逆向求输入字符串的代码：

```
var a = [118, 104, 102, 120, 117, 108, 119, 124, 48,123,101,120];
var s="";
for (i = 0; i < a.length; i++) {
  s += String.fromCharCode(a[i] - 3);
}
```

console.log("结果是: %s",s);
运行结果:



得到输入的字符串就是security-xbu
将字符串输入到Token输入框中



得到flag: RenIbyd8Fgg5hawvQm7TDQ