第4章 Web攻击及防御技术

—— CSRF攻击及防御

跨站请求伪造攻击

1 CSRF是什么？

CSRF（Cross-site request forgery）

中文名称：跨站请求伪造

也被称为：one click attack/session riding

缩写为：CSRF/XSRF

CSRF曾被列为互联网20大安全隐患之一，是一种对网站的恶意利用。

CSRF是什么？

- Cross-Site Request Forgery跨站请求伪造。
  - 1、跨站点的请求；2、请求是伪造的。
- 被攻击者的浏览器被迫向目标站点发起了伪造的请求，这个过程会带上被攻击者的身份验证标识（session）以通过目标站点的验证。从而借用被攻击者在目标站点上的权限进行一系列不被期望的操作。

利用用户已登录的身份，在用户毫不知情的情况下，

以用户的名义完成非法操作

CSRF可以做什么

你可以这么理解CSRF攻击：

攻击者盗用了你的身份，以你的名义发送恶意请求。 ------借刀杀人

CSRF能够做的事情包括：

以你名义发送邮件，发消息，盗取你的账号，甚至于购买商品，虚拟货币转账.....

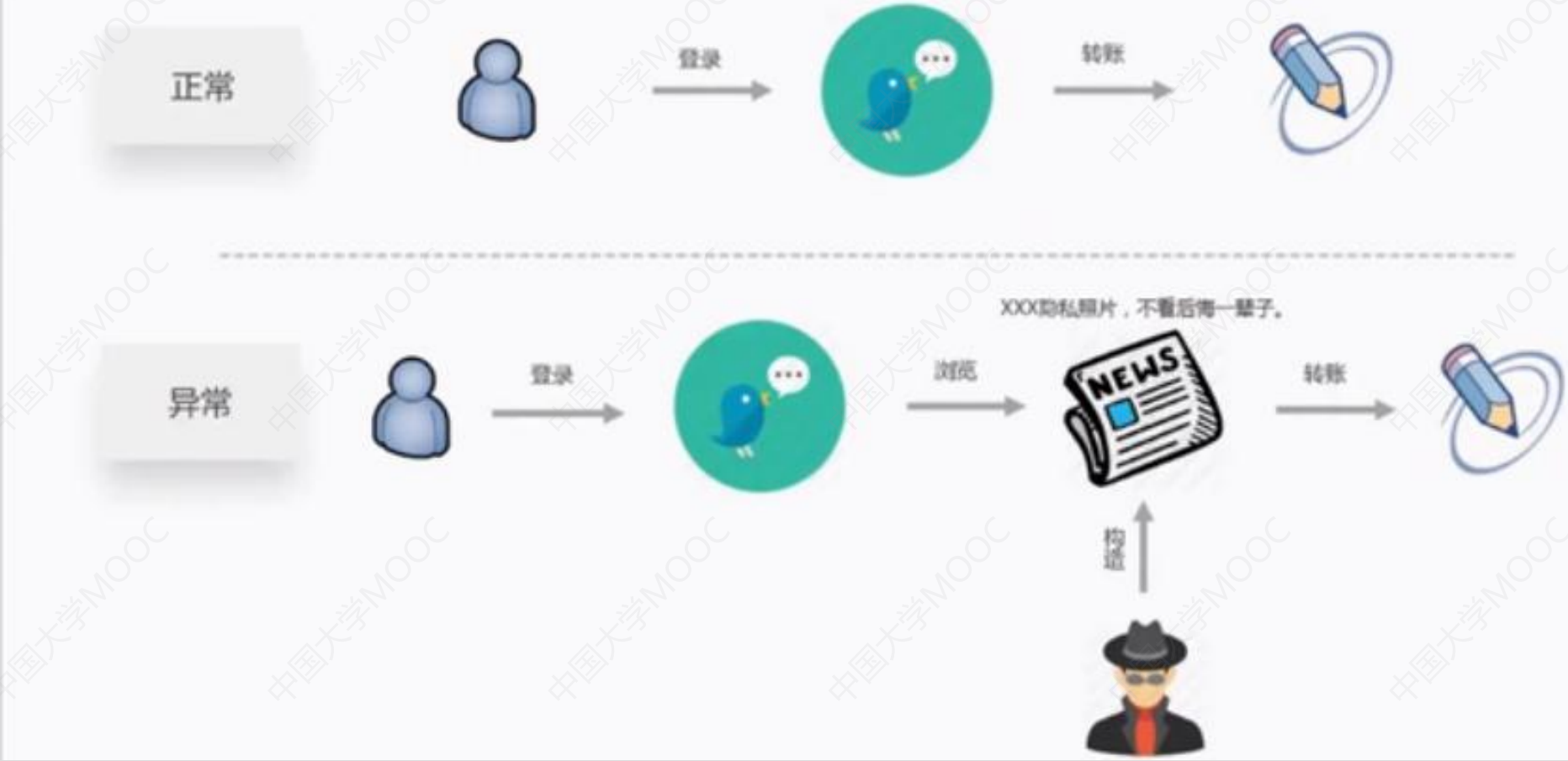
造成的问题包括：

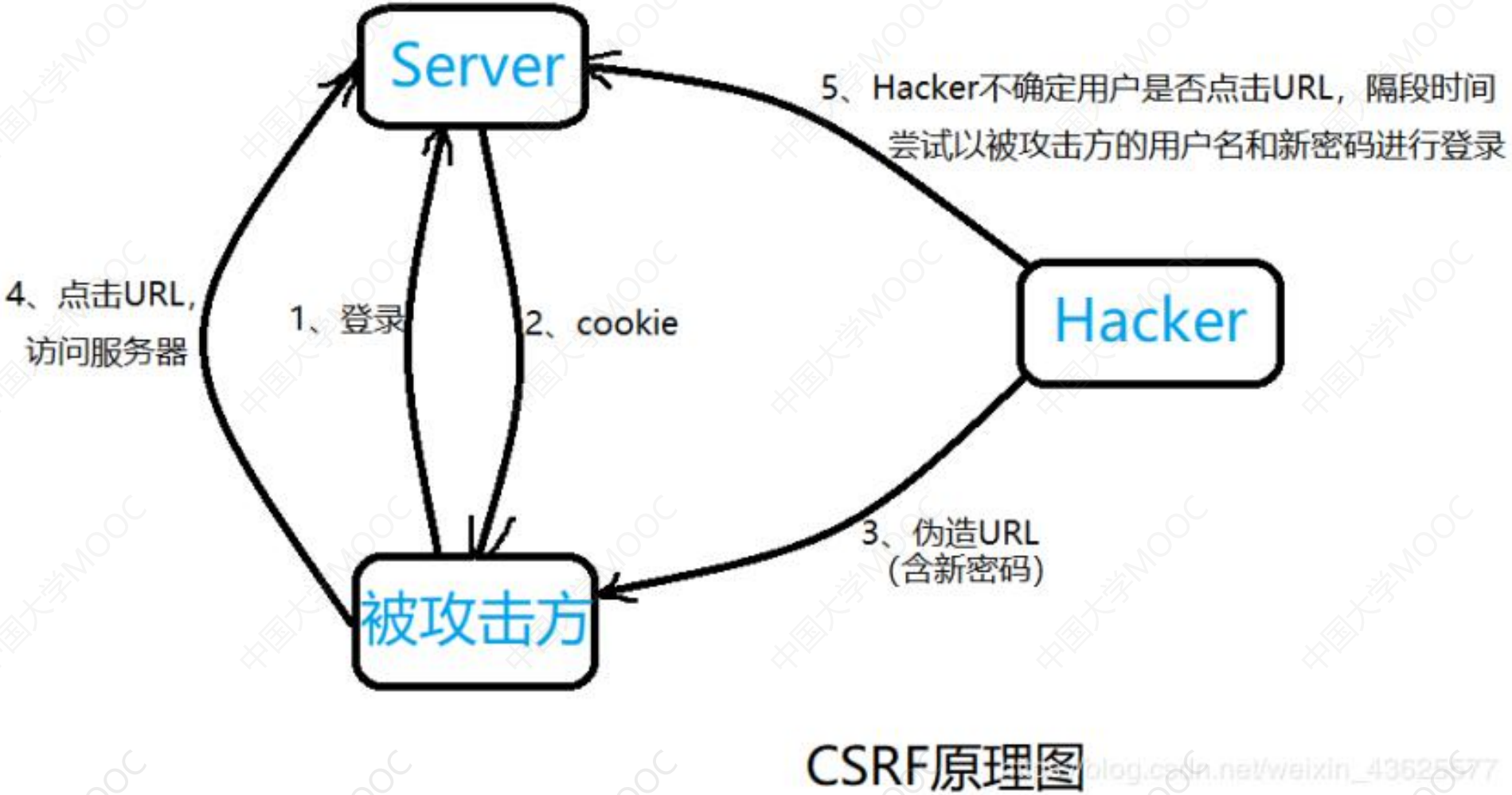
个人隐私泄露以及财产安全

谁动了我的“奶酪”

Who moved my cheese

云课堂

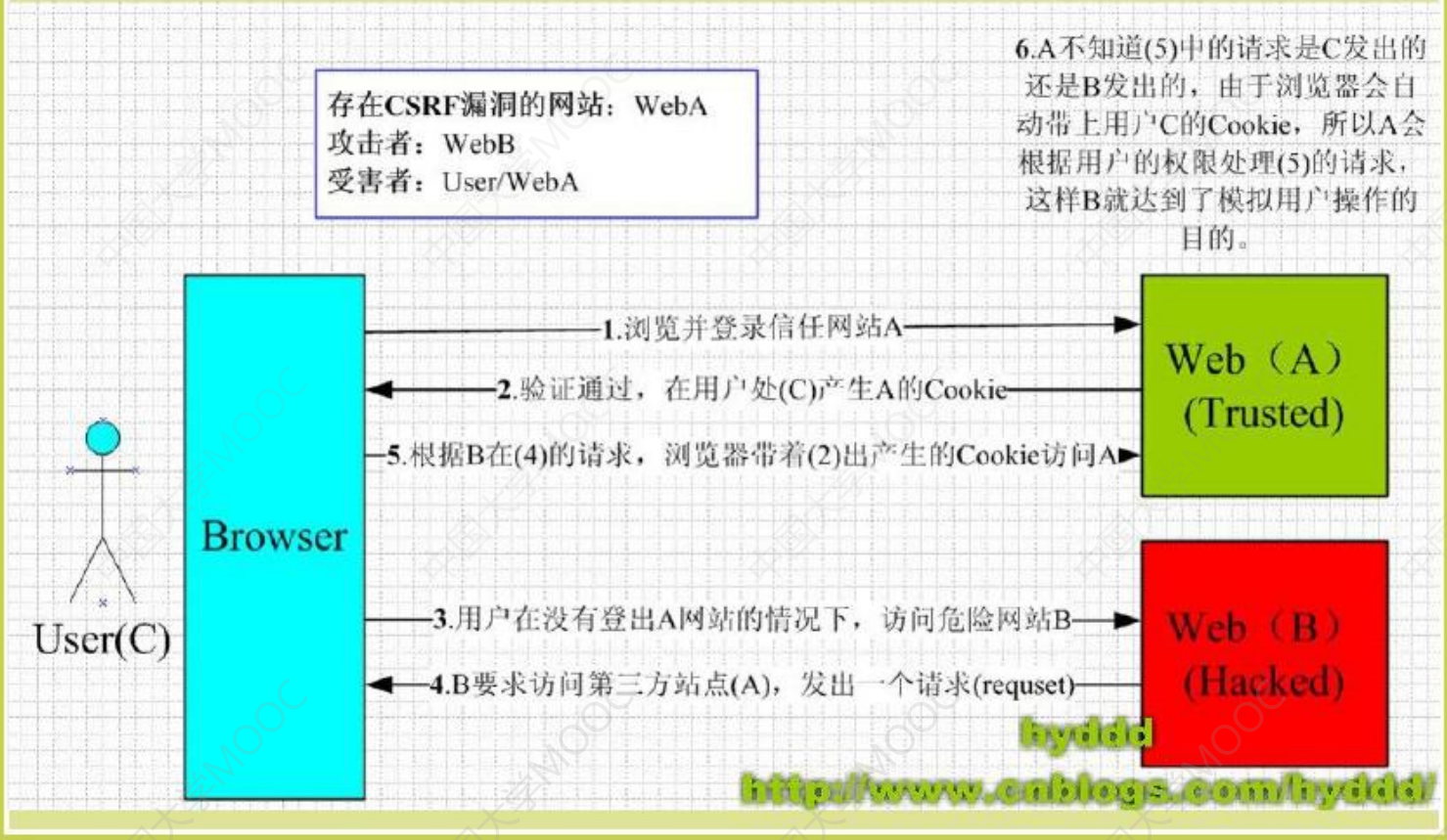



Server
5、Hacker不确定用户是否点击URL，隔段时间
尝试以被攻击方的用户名和新密码进行登录
4、点击URL，
访问服务器
1、登录
2、cookie
Hacker
3、伪造URL
(含新密码)
被攻击方

CSRF原理图

CSRF的原理

下图简单阐述了CSRF攻击的思想：



CSRF的原理

从上图可以看出，要完成一次CSRF攻击，受害者必须依次完成两个步骤：

1.登录受信任网站A，并在本地生成Cookie。

2.在不登出A的情况下，访问危险网站B。

看到这里，你也许会说：“如果我不满足以上两个条件中的一个，我就不会受到CSRF的攻击”。是的，确实如此，但你不能保证以下情况不会发生：

1.你不能保证你登录了一个网站后，不再打开一个tab页面并访问另外的网站。

2.你不能保证你关闭浏览器了后，你本地的Cookie立刻过期，你上次的会话已经结束。（事实上，关闭浏览器不能结束一个会话，但大多数人都会错误的认为关闭浏览器就等于退出登录/结束会话了.....）

3.上图中所谓的攻击网站，可能是一个存在其他漏洞的可信任的经常被人访问的网站。

CSRF原理分析
The principle of CSRF

云课堂

用户可以从自己账户转账

黑客不能从他人账户转账

黑客构造恶意网页

用户主动访问恶意网页，转账到黑客账户

CSRF原理分析
The principle of CSRF

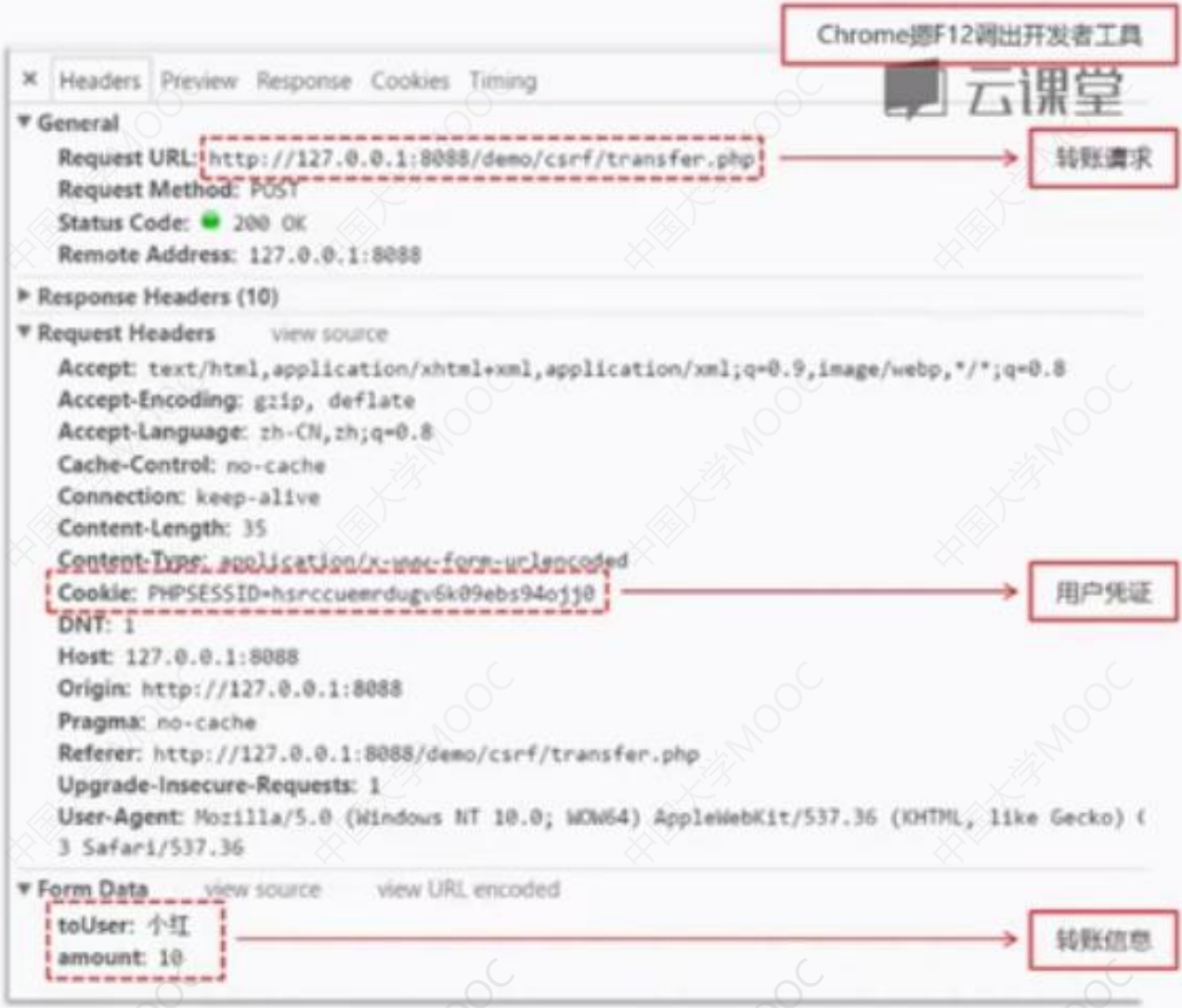
云课堂

CSRF原理分析
The principle of CSRF

通常Cookie当中会存放用户凭证信息

浏览器在发送任何请求时，会自动带上已有的cookie

通过Cookie识别用户身份后，执行转账操作



CSRF原理分析
The principle of CSRF

构造攻击代码

打开页面自动完成转账

CSRF的原理 示例一

银行网站A，它以GET请求来完成银行转账的操作，如：
http://www.mybank.com/Transfer.php?toBankId=11&money=1000

危险网站B，它里面有一段HTML的代码如下：
<img src=http://www.mybank.com/Transfer.php?toBankId=11&money=1000>

首先，你登录了银行网站A，然后访问危险网站B，这时你会发现你的银行账户少了1000块.....

原因是银行网站A违反了HTTP规范，使用GET请求更新资源。在访问危险网站B之前，你已经登录了银行网站A，而B中的<img>以GET的方式请求第三方资源（这里的第三方就是指银行网站了，原本这是一个合法的请求，但这里被不法分子利用了），所以你的浏览器会带上你的银行网站A的Cookie发出Get请求，去获取资源“http://www.mybank.com/Transfer.php?toBankId=11&money=1000”，结果银行网站服务器收到请求后，认为这是一个更新资源操作（转账操作），所以就立刻进行转账操作.....

CSRF的原理 示例二

银行网站A决定改用POST请求完成转账操作

银行网站A的WEB表单如下：

```
<form action="Transfer.php" method="POST">
        <p>ToBankId: <input type="text" name="toBankId" /></p>
        <p>Money: <input type="text" name="money" /></p>
        <p><input type="submit" value="Transfer" /></p>
</form>
```

后台处理页面Transfer.php如下：

```
<?php
        session_start();
        if (isset($_REQUEST['toBankId']&&isset($_REQUEST['money']))
        {
            buy_stocks($_REQUEST['toBankId'],$_REQUEST['money']);
        }
?>
```

CSRF的原理 示例二

危险网站B，仍然只是包含那句HTML代码：

```
<img src=http://www.mybank.com/Transfer.php?toBankId=11&money=1000>
```

和示例1中的操作一样，你首先登录了银行网站A，然后访问危险网站B，结果.....和示例1一样，你再次没了1000块

这次事故的原因是：银行后台使用了$_REQUEST去获取请求的数据，而$_REQUEST既可以获取GET请求的数据，也可以获取POST请求的数据，这就造成了在后台处理程序无法区分这到底是GET请求的数据还是POST请求的数据。

在PHP中，可以使用$_GET和$_POST分别获取GET请求和POST请求的数据。在JAVA中，用于获取请求数据request一样存在不能区分GET请求数据和POST数据的问题。

CSRF的防御 Cookie Hashing

Cookie Hashing(所有表单都包含同一个伪随机值):

这可能是最简单的解决方案了，因为攻击者不能获得第三方的Cookie(理论上)，所以表单中的数据也就构造失败了

```
<?php
//构造加密的Cookie信息
$value = “DefenseSCRF”;
setcookie(“cookie”, $value, time()+3600);
?>
```

在表单里增加Hash值，以认证这确实是用户发送的请求

```
<?php
    $hash = md5($_COOKIE['cookie']);
?>
<form method="POST" action="transfer.php">
    <input type="text" name="toBankId">
    <input type="text" name="money">
    <input type="hidden" name="hash" value="<?=$hash;?>">
    <input type="submit" name="submit" value="Submit">
</form>
```

CSRF的防御 Cookie Hashing

然后在服务器端进行Hash值验证

```
<?php
    if(isset($_POST['check']))
    {
        $hash = md5($_COOKIE['cookie']);
        if($_POST['check'] == $hash)
        {
            doJob();
        }
        else {      //...    }
    }
    else {      //...    }
?>
```

这个方法已经可以杜绝99%的CSRF攻击了，那还有1%呢....由于用户的Cookie很容易由于网站的XSS漏洞而被盗取，这就是另外的1%。一般的攻击者看到有需要算Hash值，基本都会放弃，但也会有特殊情况，所以如果需要100%的杜绝，这个不是最好的

CSRF的防御 验证码

验证码

这个方案的思路是：用户每次提交都需要在表单中填写一个图片上的随机字符串。

这个方案可以完全解决CSRF，但个人觉得在易用性方面似乎不是太好，还有听闻是验证码图片的使用涉及了一个被称为MHTML的Bug，可能在某些版本的微软IE中受影响。

谢谢各位！