Web 系统测试

4.13 渗透测试—跨站点请求伪造

河北师范大学软件学院
Software College of Hebei Normal University

内容回顾

➢浏览器安全

●同源策略

- 两个页面地址中的协议，域名(或IP)，子域名，端口号一致，则表示同源

●沙箱策略

- 泛指“资源隔离类模块”的代名词

内容回顾

- 恶意网址拦截
  - 浏览器周期性地从服务器端获取一份最新的恶意网址黑名单，如果用户上网访问的网址存在于此黑名单中，浏览器就会弹出一个警告页面
- 其他安全策略
  - IE8中推出了XSS Filter功能
  - Firfox4 推出了Content Security Policy（内容安全政策）

内容回顾

- 点击劫持概述
  - 攻击者使用一个透明的、不可见的iframe,覆盖在一个网页上,然后诱使用户在该网页上进行操作，此时用户将在不知情的情况下点击透明的iframe页面。通过调整iframe的位置，可以诱使用户恰好点击在iframe页面的一些功能性按钮上
- Flash点击劫持
  - 攻击者通过Flash构造出点击劫持
- 图片覆盖攻击
  - 通过调整图片的style使得图片能够覆盖在他所指定的任意位置

内容回顾

➢拖拽劫持与数据窃取

●诱使用户从隐藏不可见的iframe中拖拽出攻击者希望得到的数据，然后放到攻击者能控制的另外一个页面中，从而窃取数据

目 录

CSRF概述

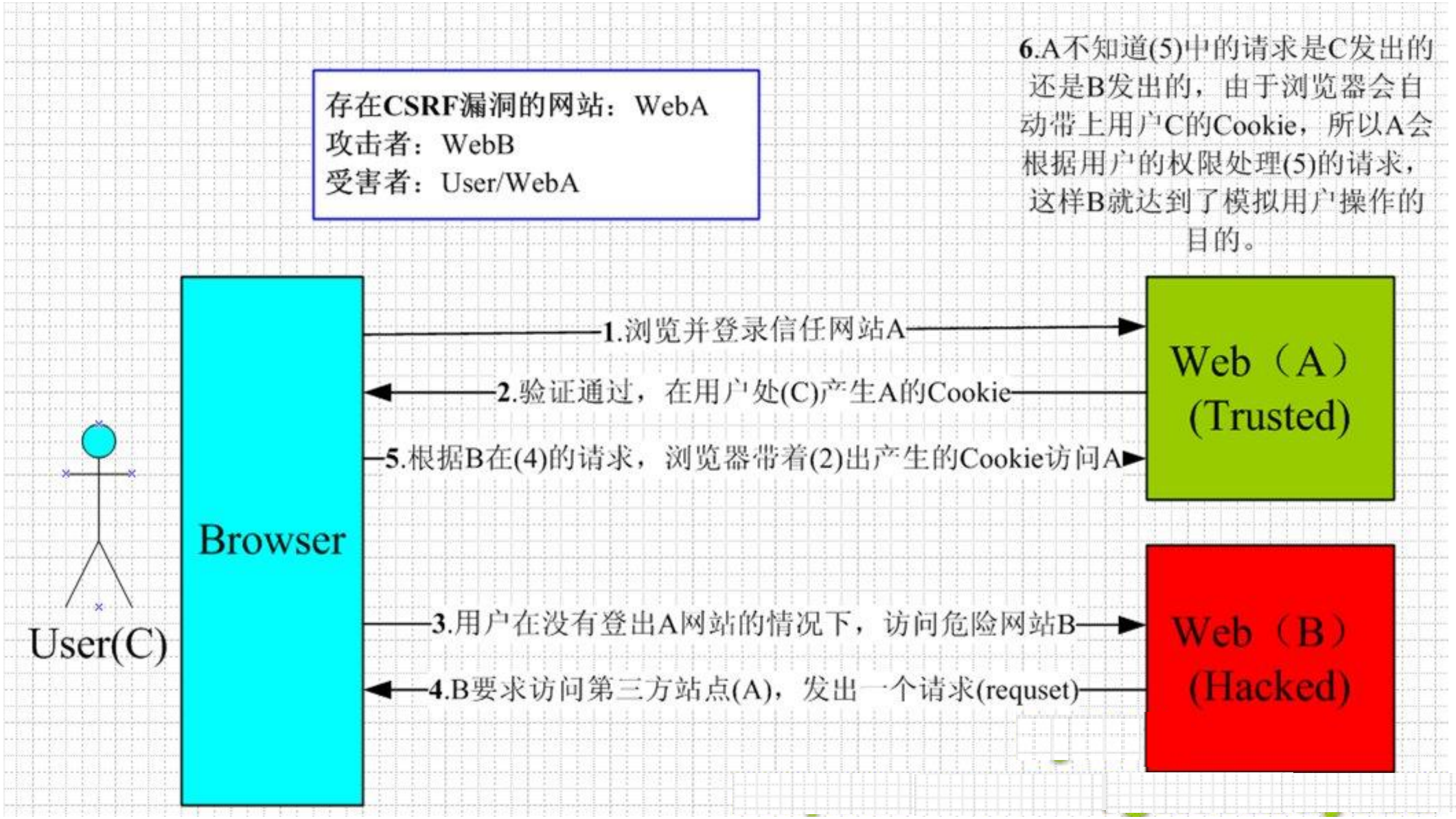
➢什么是CSRF

- ●全名：Cross Site Request Forgery（跨站点请求伪造）
- ●攻击者通过盗用身份，并用盗用来的身份发送恶意请求

➢CSRF能够做什么？

- ●以盗用者身份发送邮件，发消息，盗取账号，甚至用于购买商品，虚拟货币转账等等
- ● 造成的问题包括：个人隐私泄露以及财产安全

CSRF概述—CSRF的原理



CSRF概述

- 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。

- 如果不满足以上两个条件中的一个，就不会受到CSRF的攻击

CSRF概述

- 能不能保证以下情况不会发生：

1.不能保证登录了一个网站后，不再打开一个tab页面并访问另外的网站

2. 不能保证关闭浏览器后，本地的Cookie立刻过期，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

目 录

CSRF攻击过程

➢银行转账被攻击过程

- 银行网站A，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>

CSRF攻击过程

- 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元

➢ 为什么会这样呢？

- 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的之前，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），所以浏览器会带上银行网站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攻击过程

- 为了杜绝上面的问题，银行如果改用POST请求完成转账操作是否可行？
- 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>
```

CSRF攻击过程

- 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攻击过程

- 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>

- 结果：与如上get的操作一样，仍然被转走1000元

CSRF攻击过程

➢什么原因

- 首先登录了银行网站A，然后访问危险网站B，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攻击过程

➢如果获取请求数据的方法改了，能否解决问题

●改用$_POST，只获取POST请求的数据，后台处理页面

Transfer.php代码如下：

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

CSRF攻击过程

- 网站B也修改代码
- 如果用户继续上面的操作，结果将仍然是上面的结果

```
<html>
    <head>
        <script type="text/javascript">
            function steal()
            {
                    iframe = document.frames["steal"];
                    iframe.document.Submit("transfer");
            }
        </script>
    </head>

    <body onload="steal()">
        <iframe name="steal" display="none">
            <form method="POST" name="transfer"
action="http://www.myBank.com/Transfer.php">
                <input type="hidden" name="toBankId" value="11">
                <input type="hidden" name="money" value="1000">
            </form>
        </iframe>
    </body>
</html>
```

CSRF攻击过程

➢CSRF攻击机制？

- 上面的3种攻击模式，其实可以看出，CSRF攻击是源于Web的隐式身份验证机制！Web的身份验证机制虽然可以保证一个请求是来自于某个用户的浏览器，但却无法保证该请求是用户批准发送的

目 录

CSRF的防御

- 怎样防御CSRF攻击？
  - CSRF的防御可以从服务端和客户端两方面着手
  - 防御效果是从服务端着手效果比较好
  - 现在一般的CSRF防御也都在服务端进行

CSRF的防御

➢服务端进行CSRF防御

●服务端的CSRF方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数

Cookie Hashing

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

CSRF的防御

➢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

➢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 {
         //...
        }
 ?>
```

CSRF的防御

- 方法二：验证码
  - 思路：每次用户提交都需要用户在表单中填写一个图片上的随机字符串

CSRF的防御

➢方法三：One-Time Tokens(不同的表单包含一个不同的伪随机值)

- 在实现One-Time Tokens时，需要注意一点：“并行会话的兼容”。如果用户在一个站点上同时打开了两个不同的表单，CSRF保护措施不应该影响到他对任何表单的提交
- 如果每次表单被装入时站点生成一个伪随机值来覆盖以前的伪随机值将会发生什么情况?
  - 用户只能成功地提交他最后打开的表单，因为所有其他的表单都含有非法的伪随机值。必须小心操作以确保CSRF保护措施不会影响选项卡式的浏览或者利用多个浏览器窗口浏览一个站点

CSRF的防御

➢令牌生成函数(gen_token())

```php
<?php
  function gen_token() {
       $token = md5(uniqid(rand(), true));
       return $token;
  }
```

CSRF的防御

- 然后是Session令牌生成函数(gen_stoken()):

```
<?php
        function gen_stoken() {
                $pToken = "";
                if($_SESSION[STOKEN_NAME]  == $pToken){
                        //没有值，赋新值
                        $_SESSION[STOKEN_NAME] = gen_token();
                }
                else{
                        //继续使用旧的值
                }
        }
    ?>
```

CSRF的防御

➢WEB表单生成隐藏输入域的函数：

```
<?php
        function gen_input() {
                gen_stoken();
                echo "<input type=\"hidden\" name=\"" . FTOKEN_NAME . "\"
                        value=\"" . $_SESSION[STOKEN_NAME] . "\"> ";
        }
    ?>
```

CSRF的防御

➢WEB表单结构

```
<?php
        session_start();
        include("functions.php");
    ?>
    <form method="POST" action="transfer.php">
        <input type="text" name="toBankId">
        <input type="text" name="money">
        <? gen_input(); ?>
        <input type="submit" name="submit" value="Submit">
    </form>
```

➢服务端核对令牌

内容总结

- CSRF概述
- CSRF攻击过程
- CSRF的防御

Question