Web 系统测试

4.6 Web安全测试—跨站脚本（XSS）漏洞

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

目 录

- 什么是XSS
- XSS原理解析
- 怎样测试XSS漏洞

什么是XSS

- XSS又叫CSS（Cross Site Scripting），即跨站脚本攻击，指攻击者在网页中嵌入客户端脚本，通常是JavaScript编写的恶意代码，当用户使用浏览器被嵌入恶意代码的网页时，恶意代码将会在用户的浏览器上执行
- 产生的原因
  - Web应用程序对用户的输入过滤不严而产生的

认识XSS

- XSS（cross-site script）跨站脚本自1996年诞生以来，一直被OWASP（open web application security project）评为十大安全漏洞中的第二威胁漏洞。也有黑客把XSS当做新型的“缓冲区溢出攻击”，而JavaScript是新型的shellcode
- 2011年6月份，国内最火的信息发布平台“新浪微博”爆发了XSS蠕虫攻击，仅持续16分钟，感染用户近33000个，危害十分严重
- XSS最大的特点就是能注入恶意的代码到用户浏览器的网页上，从而达到劫持用户会话的目的

XSS的危害

1. 网络钓鱼，包括窃取各类用户账号
2. 窃取用户cookie
3. 窃取用户浏览会话
4. 强制弹出广告页面、刷流量
5. 网页挂马
6. 提升用户权限，进一步渗透网站
7. 传播跨站脚本蠕虫等

XSS类型

- 反射型XSS
- 存储型XSS
- DOM XSS

反射型XSS

➢反射型跨站脚本也称做非持久型、参数型跨站脚本、这类型的脚本是最常见的，也是使用最为广泛的一种

●可以将恶意的脚本附加到URL地址的参数

- 例如：

  http://www.xxcc.com/search.php?key=“><script>alert(“xss”)</script>

- 一般使用将构造好的URL发给受害者，使受害者点击触发，而且只执行一次，非持久化

●或者将恶意脚本附加到带参数的输出函数中

反射型XSS—将恶意代码附着在参数中实例


```
<html>
    <body>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>XSS</title>
        </head>
        <form action="" method="get">
            <input type="text" name="input">
            <input type="submit">
        </form>
        <br>
        <?php
        $XssReflex = $_GET['input'];
        echo 'output:<br>'.$XssReflex;
        ?>
    </body>
</html>
```

反射型XSS

- 将此文件放在Apache根目录下
- 使用火狐浏览器打开
- 在输入框中输入1，hello
- 在输入框中输入：

  <script>alert('xss')</script>

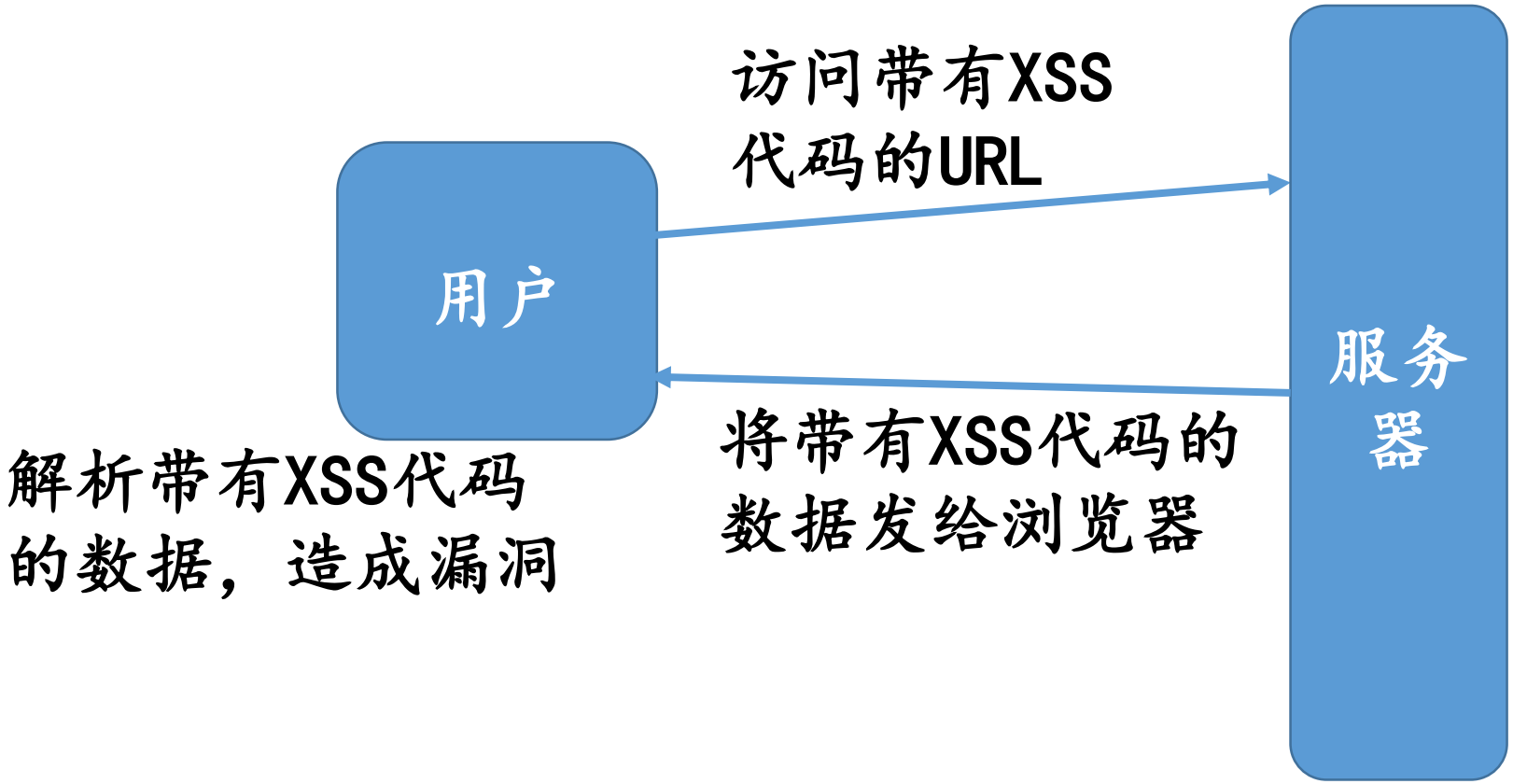
  查看弹窗，会将“xss”弹出

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4  <title>XSS</title>
5  </head>
6  <body>
7  <form action="" method="get">
8  <input type="text" name="input">
9  <input type="submit">
10 </form>
11 <br>
12 output:<br><script>alert('xss')</script>
13 </body>
14 </html>
```

反射型XSS

- 思考：若script中间不是alert 语句，是其他语句
  - 比如：document.cookie ,会怎样
  - 是其他脚本，会怎样？

反射型XSS

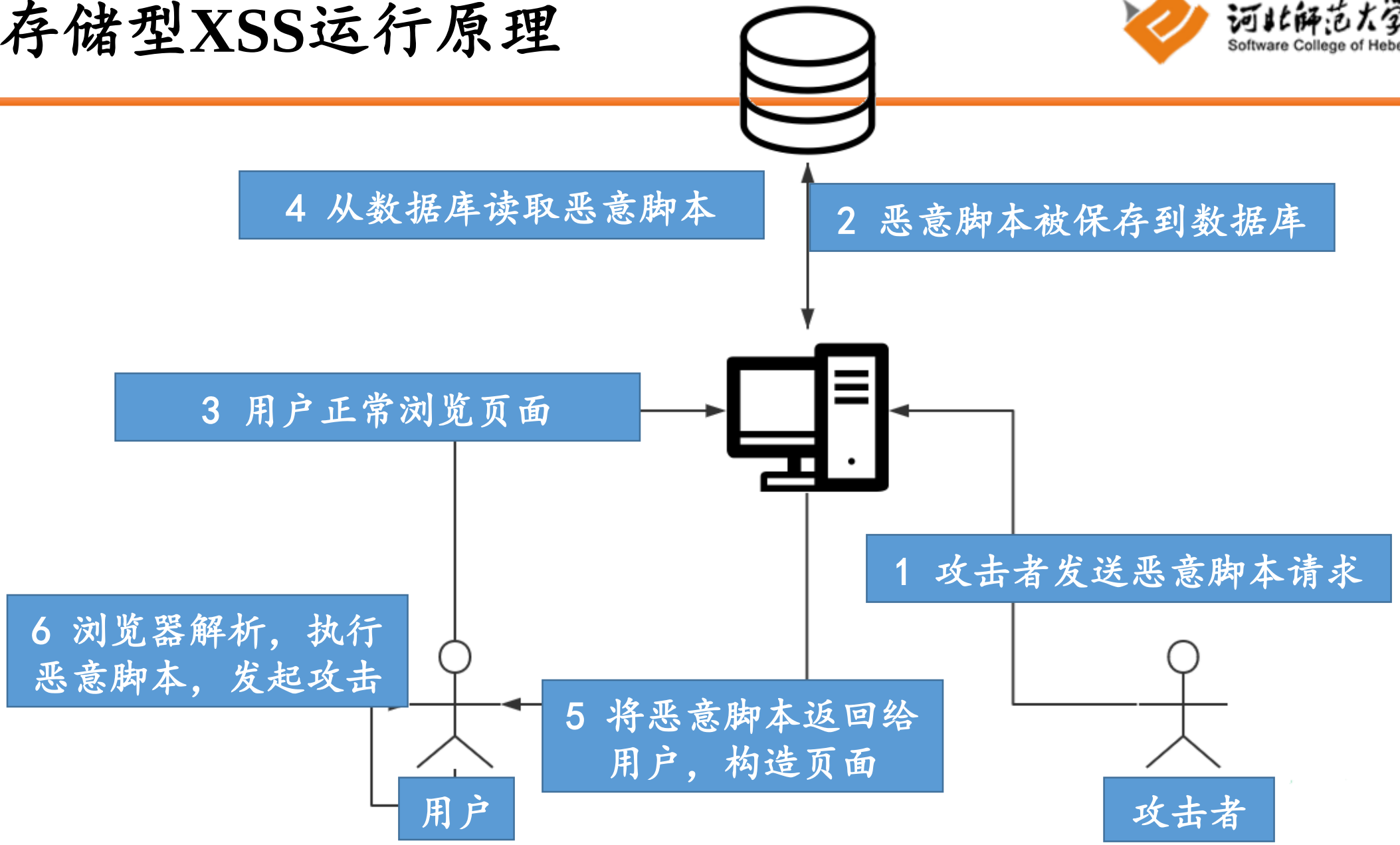


- 这个过程像一次反射，所以称为反射型XSS

存储型XSS

- 什么是存储型XSS
  - 当用户提交一段XSS代码后，被服务器端接收并存储，当攻击者再次访问某个页面时，这段XSS代码被程序读出来响应给浏览器，造成XSS跨站攻击
- 什么情况容易出现存储型XSS
  - 运行用户存储数据的Web应用程序

存储型XSS运行原理


4 从数据库读取恶意脚本
2 恶意脚本被保存到数据库
3 用户正常浏览页面
1 攻击者发送恶意脚本请求
6 浏览器解析，执行恶意脚本，发起攻击
5 将恶意脚本返回给用户，构造页面
用户
攻击者

存储型XSS—举例


```
<html>
    <head>
    <title>XssStorage</title>
    </head>
    <body>
    <h2>Message Board<h2>
    <br>
    <form action="XssStorage.php" method="post">
    Message:<textarea id='Mid' name="desc"></textarea>
    <br>
    <br>
    Subuser:<input type="text" name="user"/><br>
    <br>
    <input type="submit" value="submit" onclick='loction="XssStorage.php"'/>
    </form>
```

存储型XSS —举例


```
16          <?php
17          if(isset($_POST['user'])&&isset($_POST['desc'])){
18          $log=fopen("sql.txt","a");
19          fwrite($log,$_POST['user']."\r\n");
20          fwrite($log,$_POST['desc']."\r\n");
21          fclose($log);
22          }
23
24          if(file_exists("sql.txt"))
25          {
26          $read= fopen("sql.txt",'r');
27          while(!feof($read))
28          {
29              echo fgets($read)."</br>";
30          }
31          fclose($read);
32          }
33          ?>
34          </body>
35      </html>
```

存储型XSS —举例

➢实验：

- 在输入框中，输入普通文字
- 在输入框中，输入JS脚本
  - 如：<script>alert('xss')</script>
  - 重启浏览器之后再加载该页面，页面依然会弹窗,这是因为恶意代码已经写入数据库中，每当有人访问该页面时，恶意代码就会被加载执行

存储型XSS —举例

- 查看网页html代码

view-source:http://192.168.2.102:8032/XssStor

Search

```
<html>
<head>
<title>XssStorage</title>
</head>
<body>
<h2>Message Board<h2>
<br>
<form action="XssStorage.php" method="post">
Message:<textarea id='Mid' name="desc"></textarea>
<br>
<br>
Subuser:<input type="text" name="user"/><br>
<br>
<input type="submit" value="submit" onclick='loction="XssStorage.php"'/>
</form>
dfads
</br>adfd
</br>abc
</br><script>alert("hello")</script>
</br>scripttest
</br><script>alert("xss")</script>
</br></br>
</body>
</html>
```

存储型XSS

- 存储型XSS漏洞，一次提交之后，每当有用户访问这个页面都会受到XSS攻击，危害巨大。

DOM XSS

- **DOM： Document Object Model**
- **DOM XSS**
  - 使用DOM可以允许程序和脚本动态地访问和更新文档的内容、结构和样式

DOM XSS


```
<script>
      var tmp = document.URL;          //获取url
     var index = document.URL.indexOf("content=") + 4;
      var par = temp.substring(index);
      document.write(decodeURI(par));    //输入获取内容
</script>
```

如果输入的内容中包含 <script>alert(/xss)</script>就会产生XSS漏洞

检测XSS

➢检测方式

- ●手工检测
- ●自动检测

检测XSS

- 手工检测
  - 输入一些敏感字符，如“<、>、’、（）”，提交后查看HTML源代码，看这些是否被转义
- 全自动检测XSS，借助于扫描工具
  - awvs
  - netsparke
  - appscan
  - burpsuit
  - xsser
  - xsscrapy
  - brutexssr
  - OWASP Xenotix

XSS漏洞防范

➢反射型XSS漏洞防范

一．PHP中直接输出html的，可以采用以下方法进行过滤

1. htmlspecialchars函数
2. htmlentities函数
3. HTMLPurifier.auto.php插件
4. removexss函数

反射型XSS漏洞防范

二 . PHP输出到JS代码中，或者开发Json API的，则需要前端在JS中进行过滤：

1. 尽量使用innerText(IE)和textContent(Firefox),也就是jQuery的text()来输出文本内容

2. 必须要用innerHTML等函数，则需要做类似php的htmlspecialchars的过滤

反射型XSS漏洞防范

三.其它的通用的补充性防御手段

1　在输出html时，加上Content Security Policy的Http Header

（作用：可以防止页面被XSS攻击时，嵌入第三方的脚本文件等）

（缺陷：IE或低版本的浏览器可能不支持）

2.在设置Cookie时，加上HttpOnly参数

（作用：可以防止页面被XSS攻击时，Cookie信息被盗取，可兼容至IE6）

（缺陷：网站本身的JS代码也无法操作Cookie，而且作用有限，只能保证Cookie的安全）

3.在开发API时，检验请求的Referer参数

（作用：可以在一定程度上防止CSRF攻击）

（缺陷：IE或低版本的浏览器中，Referer参数可以被伪造）

反射型XSS漏洞防范


```
<html>
    <body>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>XSS</title>
        </head>
        <form action="" method="get">
            <input type="text" name="input">
            <input type="submit">
        </form>
        <br>
        <?php
        $XssReflex = $_GET['input'];
        echo 'output:<br>'.htmlentities($XssReflex);  #仅在这里对变量 $XssReflex 做了处理.
        ?>
    </body>
</html>

<span style="font-size:18px;"><meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
</span>
```

反射型XSS漏洞防范

➢查看页面源代码

```
<html>
    <body>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
            <title>XSS</title>
        </head>
        <form action="" method="get">
            <input type="text" name="input">
            <input type="submit">
        </form>
        <br>
        output:<br>&lt;script&gt;alert(&quot;abc&quot;)&lt;/script&gt;
    </body>
</html>

<span style="font-size:18px;"><meta http-equiv="Content-Type" content="text/html;charset=utf
</span>
```

反射型XSS漏洞防范

- **htmlentities() :把预定义的字符 "<" （小于）和 ">" （大于）转换为 HTML 实体**

存储型XSS漏洞防范

- 存储型XSS对用户的输入进行过滤的方式和反射型XSS相同，这里我们使用htmlspecialchars()函数
- htmlspecialchars和htmlentities的区别：
  - htmlspecialchars 只转义 & 、" 、' 、< 、> 这几个html代码
  - htmlentities 却会转化所有的html代码，连同里面的它无法识别的中文字符也会转化

存储型XSS漏洞防范


```
<span style="font-size:18px;"><meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
    <html>
    <head>
    <title>XssStorage</title>
    </head>
    <body>
    <h2>Message Board<h2>
    <br>
    <form action="Xss_htmlspecialchars_Storage.php" method="post">
    Message:<textarea id='Mid' name="desc"></textarea>
    <br>
    <br>
    Subuser:<input type="text" name="user"/><br>
    <br>
    <input type="submit" value="submit" onclick='loction="XssStorage.php"'/>
    </form>
```

存储型XSS漏洞防范


```php
<?php
if(isset($_POST['user'])&&isset($_POST['desc'])){
$log=fopen("sqlStorage.txt","a");
fwrite($log,htmlspecialchars($_POST['user'])."\r\n"); # 在此对用户输入数据$_POST['user']进行过滤
fwrite($log,htmlspecialchars($_POST['desc'])."\r\n"); # 在此对用户输入数据$_POST['desc']进行过滤
fclose($log);
}

if(file_exists("sqlStorage.txt"))
{
$read= fopen("sqlStorage.txt",'r');
while(!feof($read))
{
    echo fgets($read)."</br>";
}
fclose($read);
}
?>
</body>
</html></span>
```

存储型XSS漏洞防范

- 访问该页面，并在表单中输入脚本
  - 如：<script>alert (‘abc’)</script>
- 查看页面源代码
- 查看sqlStorage.txt文件

总结

- 什么是XSS
- XSS分类
- XSS原理解析
- XSS举例
- 怎样测试XSS漏洞
- 怎样防范XSS漏洞

练习

- 完成反射型和存储型XSS页面构造和攻击
- 尝试在zl_shop网站上进行XSS攻击
- 尝试在本地页面中填写获取用户cookie，或获取用户账户、密码，或获取用户数据库权限的JS脚本

Question