第 5 章

XSS 跨站脚本攻击

本章要点

- XSS 漏洞的原理
- XSS 漏洞的分类：反射型、存储型、DOM 型
- XSS 漏洞挖掘与绕过
- 基于靶机 XSS 攻击的测试案例
- XSS 攻击防御

XSS(Cross Site Scripting，跨站脚本)攻击自 1996 年诞生以来一直都是 OWASP 十大漏洞之一，尤其在 OWASP Top 10 2013 中 XSS 被列为排名第三的风险类别，在 OWASP Top 10 2017 中 XSS 被列为排名第七的风险类别，在 OWASP Top 10 2021 中 XSS 被纳入排名第三的“注入”风险类别。

本章将从 XSS 攻击经典案例开始，分别探讨什么是 XSS 攻击、如何检测是否存在 XSS 漏洞、如何利用 XSS 漏洞及如何防御 XSS 攻击。

5.1 XSS 漏洞原理

5.1.1 XSS 概述

跨站脚本(Cross Site Scripting)的字母缩写原为 CSS，为了避免与层叠样式表(Cascading Style Sheets)的缩写 CSS 混淆，所以跨站脚本攻击的缩写改为 XSS，即 XSS 攻击。

XSS 攻击通常是指通过利用网页开发时留下的漏洞，通过巧妙的方法注入恶意指令代码到网页，使用户加载并执行攻击者恶意制造的网页程序。这些恶意网页程序通常是 JavaScript，但实际上也可以包括 Java、VBScript、ActiveX、Flash 或某些普通的 HTML 等。攻击成功后，攻击者可能得到更高的权限(如执行一些操作)、私密的网页内容、会话信息和 Cookie 等各种用户敏感信息。

XSS 攻击是一种经常出现在 Web 应用程序中的计算机安全漏洞，是由于 Web 应用程序对用户的输入过滤不严而产生的。攻击者利用网站漏洞把恶意的脚本代码注入网页中，当其他用户浏览这些网页时，就会执行其中的恶意代码。

最早期的 XSS 攻击示例大多使用了跨站方法，即用户在浏览 A 网站时，攻击者却可以通过页面上的恶意代码访问用户浏览器中的 B 网站资源(如 Cookie 等)，从而达到攻击

的目的。但随着浏览器安全技术的进步,早期的跨站方法已经很难奏效,XSS 攻击也逐渐和“跨站”的概念没有了必然的联系。只不过由于历史习惯,XSS 这个名字一直被沿用了下来,现如今用来泛指通过篡改页面,使浏览器加载恶意代码的一种攻击方法。

2011 年 6 月,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。

关于脚本,现在大多数网站都使用 JavaScript 或 VBScript 来执行计算、页面格式化、Cookie 管理以及其他客户动作。以 JavaScript 为例,JavaScript 是一种浏览器端的脚本语言。用来在网页客户端处理与用户的交互,以及实现页面特效。比如提交表单前先验证数据合法性,减少服务器错误和压力。根据客户操作给出一些提升,让用户体验更好等。也可以实现一些页面动画。

JavaScript 程序可以检测网页中的各种事件并作出反应,也可以实现动态改变网页的 CSS 样式和结构,与页面各种元素进行交互等。

脚本是嵌入网页中的,JavaScript 代码编写在＜head＞头部信息的＜script＞＜/script＞标签中。

如弹出对话框 hello 的 JavaScript 脚本:

```
<script>alert("hello")</script>
```

XSS 漏洞一般出现在网页中的评论框、留言板、搜索框等“用户输入”的地方。XSS 漏洞形成的原因主要是 Web 服务器端没有对脚本文件(如＜script＞)进行安全过滤。

5.1.2 XSS 漏洞的攻击

1. XSS 典型案例

从一个 XSS 漏洞测试的典型例子来分析 XSS 漏洞的原理。

本章 XSS 漏洞测试都是基于 DVWA 漏洞测试靶场,DVWA 是一个基于 PHP 和 MySQL 开发的漏洞测试平台。测试环境搭建及说明详见附录 A 的内容。图 5-1 是 XSS 漏洞测试用户输入界面。

测试输入:1。

网站将输入的内容直接回显在页面,回显页面如图 5-2 所示。

继续测试是否存在 XSS 漏洞,构造测试脚本。

输入:＜script＞alert(/xss/)＜/script＞。

页面成功弹窗,对用户输入的＜script＞alert(/xss/)＜/script＞当作脚本执行了,验证了网站存在 XSS 漏洞,且为反射型 XSS 漏洞,如图 5-3 所示。

2. XSS 漏洞原理分析

按 F12 键查看网页源代码,发现浏览器成功将测试输入的＜script＞alert(/xss/)＜/script＞作为 HTML 元素解释运行,如图 5-4 所示。

分析 XSS 漏洞测试成功执行的原因,主要是 Web 服务器端没有对脚本文件(如＜script＞)进行安全过滤。

那么 XSS 攻击为什么是对客户端的攻击而不是对服务器端的攻击?

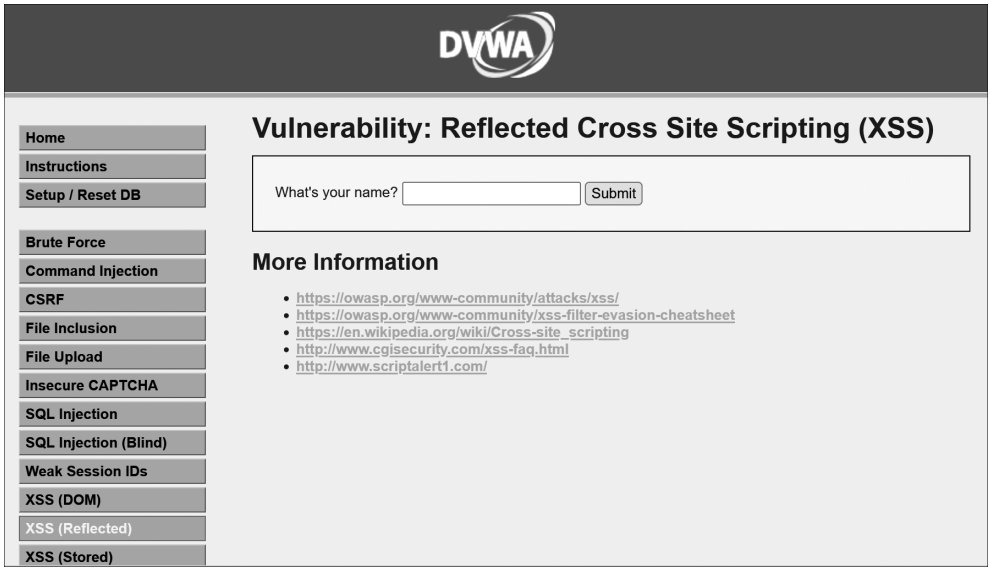

图 5-1　XSS 漏洞测试用户输入界面

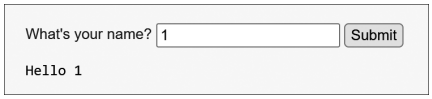

图 5-2　XSS 回显页面

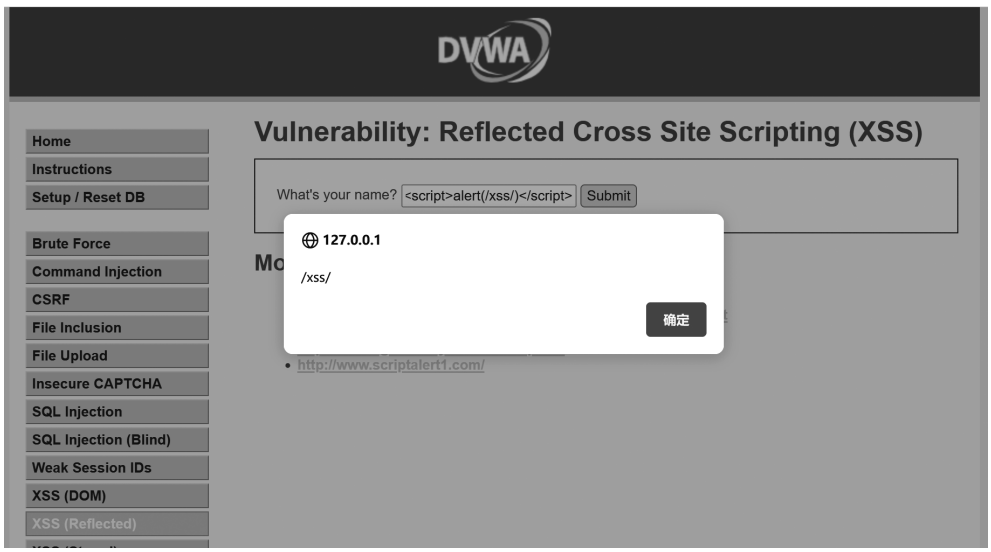

图 5-3　XSS 弹窗测试

一般早期的攻击目标都选定服务器端，因为服务器端的资源是比较丰富的，攻击服务器端是最好的。但是随着服务器端的安全已经很成熟了，有很多的产品来防护。而客户端的安全防护还在动态发展的过程中，相对来说客户端的安全比较薄弱，很容易成为被攻击的目标。如果攻击代码存储在服务器端，每个浏览的客户都会访问攻击网站，账号、密

图 5-4 XSS 测试成功执行后的网页源代码

码、Cookie 等就会泄露。

3. XSS 漏洞的危害

(1) 网络钓鱼,包括盗取各类用户账号。

(2) 窃取用户 Cookie。

(3) 窃取用户浏览记录。

(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。

5.1.3 XSS 相关知识

1. JavaScript

JavaScript(JS)是一种直译式脚本语言,它的解释器被称为 JavaScript 引擎,为浏览器的一部分,广泛用于客户端的脚本语言,最早在 HTML(标准通用标记语言下的一个应用)网页上使用,用来给 HTML 网页增加动态功能。

如果要深入研究 XSS 攻击,必须要精通 JavaScript,JavaScript 能够实现什么效果,XSS 攻击的威力就有多大。

JavaScript 还可以轻易实施下面的任何攻击:

(1) 通过 Cookie 窃取实现会话劫持。

(2) 按键记录,将所有输入的文本发送到攻击者网站。

(3) 向网页中注入链接或广告。

(4) 立即将网页重定向到恶意网站。

(5) 网站涂改。

(6) 窃取用户登录凭证。

2. Document 对象

JavaScript 中的 Document 是一个对象，从 JavaScript 一开始就存在的一个对象，它代表当前的页面(文档)。

DOM 通常用于代表在 HTML、XHTML 和 XML 中的对象，使用 DOM 可以运行程序和脚本动态地访问和更新文档的内容、结构和样式。根据 DOM 规定，HTML 文档中的每个成分都是一个节点。

如何获取一个 HTML 元素内容？

第一步，获取元素。

getElementById()：通过 id 获取元素。

第二步，获取元素内容。

.innerHTML：获取元素内容。

例如，以下 HTML 文件用于获取 p 标签的内容“Hello World”，JavaScript 的函数 getElementById()用于获取 id 的元素：myid。浏览器访问该 HTML 文件，返回界面如图 5-5 所示。

HTML 文件代码：

```
<html>
<head>
    <title>JavaScript-DOM 获取元素及内容</title>
</head>
<body>
    <p id="myid">Hello!</p>
<script>
    x=document.getElementById("myid")
    alert(/id 为 myid 元素的内容是:/+x.innerHTML);
</script>
</body>
</html>
```

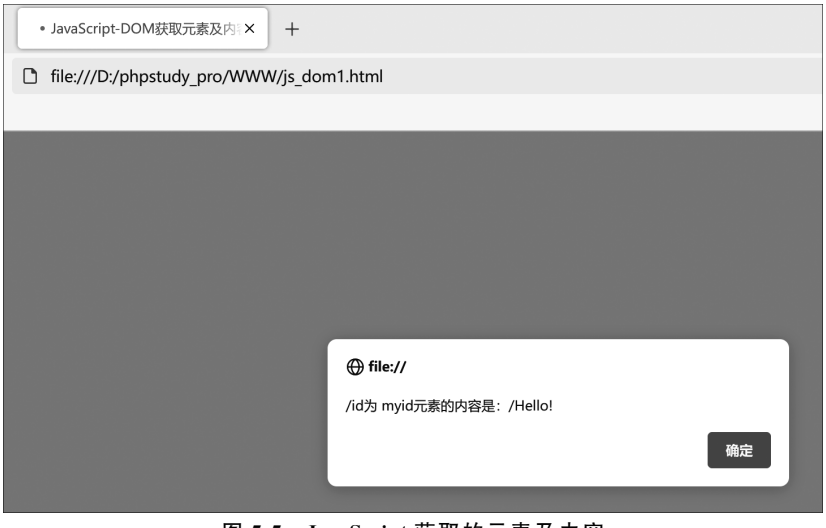

图 5-5　JavaScript 获取的元素及内容

5.2 XSS 漏洞分类

XSS 主要被分为三类,分别是反射型 XSS、存储型 XSS 和 DOM 型 XSS。

5.2.1 反射型 XSS

反射型 XSS 也称作非持久型、参数型 XSS,需要欺骗用户自己单击链接才能触发 XSS 代码,一般容易出现在搜索页面。只要用户单击特定恶意链接,服务器解析响应后,在返回的响应内出现攻击者的 XSS 代码后,被浏览器执行。一来一去,XSS 攻击脚本被 Web Server 反射给浏览器执行,所以称为反射型 XSS。反射型 XSS 大多是用来盗取用户的 Cookie 信息的。

这种类型的 XSS 是最常见的,也是使用最为广泛的一种,主要用于将恶意的脚本附加到 URL 地址的参数中。例如:

```
http://127.0.0.1/dvwa/vulnerabilities/xss_r/?name=<script>alert(/xss/)
</script>
```

一般攻击者将构造好的 URL 发给受害者,使受害者单击触发,而且只执行一次,非持久化。

反射型 XSS 的特点如下。

(1) 反射型 XSS 攻击代码不是持久性的,也就是没有保存在 Web Server 中,而是出现在 URL 地址中。

(2) 不是持久性的,那么攻击方式就不同了。一般是攻击者通过邮件、聊天软件等方式发送攻击 URL,然后用户单击来达到攻击目的。

反射型 XSS 的攻击过程如图 5-6 所示。

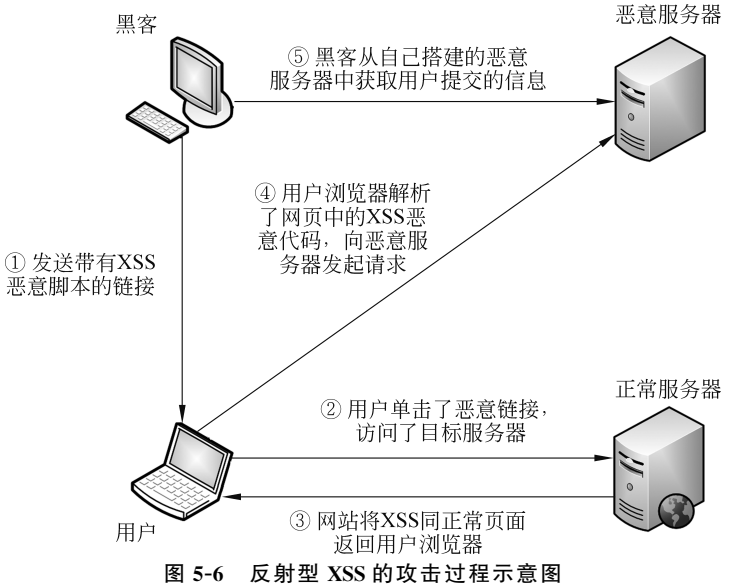

图 5-6 反射型 XSS 的攻击过程示意图

5.2.2　存储型 XSS

存储型 XSS 也称作持久型 XSS，XSS 代码是存储在 Web 服务器中的，比如在发送信息或发表文章的地方插入代码，如果没有过滤或者过滤不严，那么这些代码将存储在服务器中，用户访问该页面的时候触发代码执行。这种 XSS 比较危险，容易造成蠕虫、盗窃 Cookie。每一个访问特定页面的用户都会受到攻击。

存储型 XSS 比反射型 XSS 更具威胁性，并且可能会影响 Web 服务器的自身安全。

存储型 XSS 的特点：

(1) XSS 攻击代码存储于 Web Server 上。

(2) 攻击者一般通过网站的留言、评论、博客、日志等功能(所有能够向 Web Server 输入内容的地方)，将 XSS 攻击代码存储到 Web Server 上。

存储型 XSS 的攻击过程如图 5-7 所示。

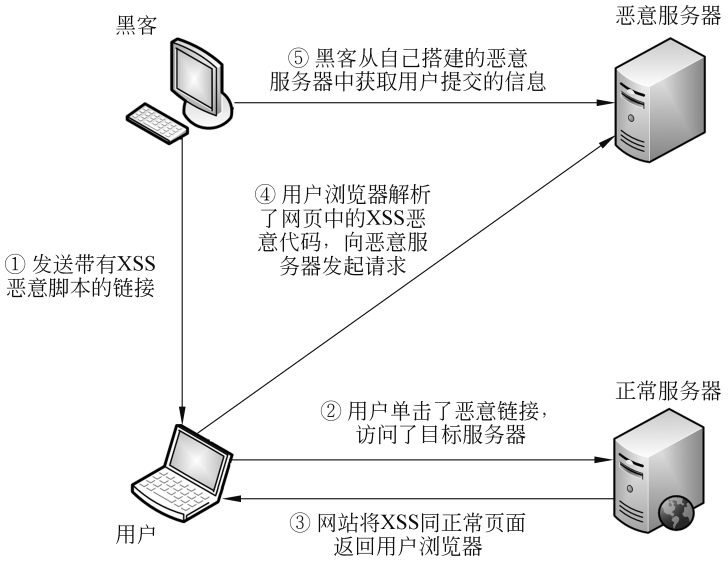

图 5-7　存储型 XSS 的攻击过程示意图

5.2.3　DOM 型 XSS

这种类型的 XSS 并非按照“数据是否保存在服务器”来划分，不经过后端的 DOM-XSS 漏洞是基于文档对象模型(Document Object Model，DOM)的一种漏洞，通过修改页面的 DOM 节点形成的 XSS 攻击，称为 DOM 型 XSS。

DOM-XSS 是通过 URL 传入参数来控制触发的，其实也属于反射型 XSS，特点是 Web Server 不参与，仅涉及浏览器的 XSS。

一般可触发 DOM 型 XSS 的属性：

(1) document.referer

(2) window.name

(3) location

(4) innerHTML

(5) document.write

(6) document.cookie

存储型、反射型和 DOM 型 XSS 攻击比较如表 5-1 所示。

表 5-1 存储型、反射型和 DOM 型 XSS 攻击比较

XSS 攻击类型	存 储 型	反 射 型	DOM 型
触发过程	(1) 黑客构造 XSS 脚本 (2) 正常用户访问携带 XSS 脚本的页面	正常用户访问携带 XSS 脚本的 URL	正常用户访问携带 XSS 脚本的 URL
数据存储	数据库	URL	URL
谁来输出	后端 Web 应用程序	后端 Web 应用程序	前端 JavaScript
输出位置	HTTP 响应中	HTTP 响应中	动态构造的 DOM 节点

5.3 XSS 漏洞挖掘与绕过测试实例

基于 Web 漏洞测试靶场,首先查看服务器端的 PHP 代码,进行代码审计及 XSS 漏洞分析,设计 XSS 漏洞测试脚本并对测试结果分析说明。

5.3.1 反射型 XSS 漏洞测试

基于 DVWA 靶场测试反射型 XSS,安全级别设置为 Medium,如图 5-8 所示。

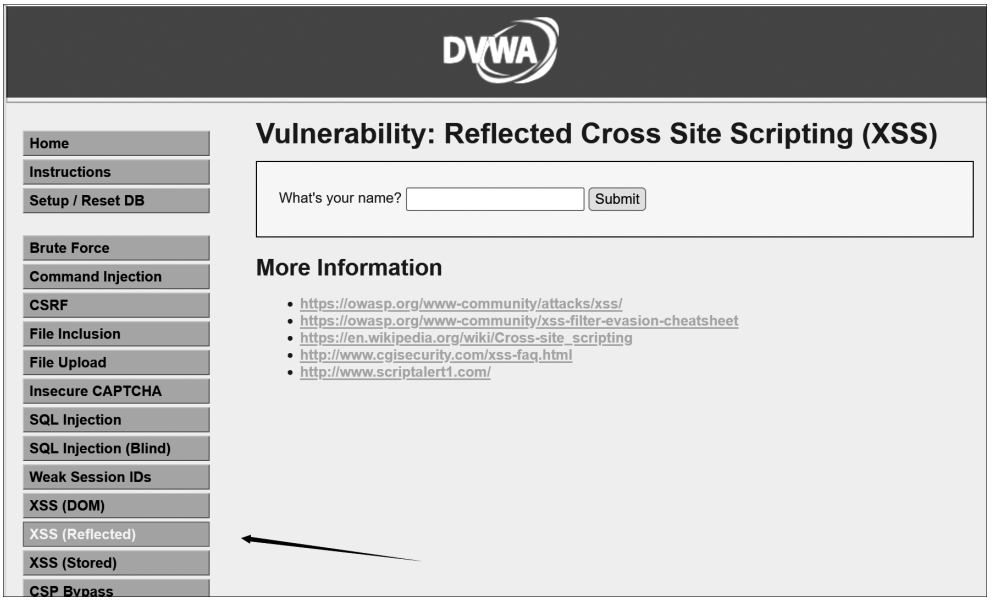

图 5-8 DVWA 测试反射型 XSS

1. 服务器端 PHP 代码

```
<?php
header ("X-XSS-Protection: 0");
//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>";
}
?>
```

2. 代码审计及 XSS 漏洞分析

上述 PHP 代码中的 str_replace()函数是用其他字符替换字符串中的一些字符(区分大小写),函数语法:str_replace(find,replace,string,count),各参数描述如下。

(1) find:在字符串 string 中要查找的值。

(2) replace:替换 find 中的值。

(3) string:被搜索的字符串。

(4) count:可选参数,对替换值进行计数的变量。

XSS 漏洞分析:

PHP 代码中的 str_replace('<script>', '', $_GET['name']),其作用是把用户输入中含有的小写字母<script>替换为空。

str_replace()函数只删除了小写的<script>标签,这种情况是很容易绕过的,可使用<script>双写绕过、大小写字母混写绕过、输入其他标签绕过等。

3. 绕过设计与测试

基于上面对代码的审计分析,设计使用<script>双写绕过或大小写字母混写绕过分别测试 XSS 漏洞。

1) <script>双写绕过测试

输入测试脚本:<sc<script>ript>alert(/xss/)</sc<script>ript>。

输入测试脚本后,返回的弹窗页面如图 5-9 所示。

函数 str_replace('<script>', '', $_GET['name'])将输入测试脚本中的<script>过滤掉,即过滤脚本<sc<script>ript>alert(/xss/)</sc<script>ript>中下画线的内容,过滤后的测试脚本的内容为<script>alert(/xss/)</script>。

按 F12 键查看源代码,发现浏览器成功将过滤后的测试脚本作为 HTML 元素解释运行,如图 5-10 所示。

2) 大小写字母混写绕过测试

输入:<ScRiPt>alert(/xss/)</ScRiPt>。

函数 str_replace('<script>', '', $_GET['name'])只过滤掉小写<script>,同样会返回如图 5-9 所示的弹窗页面,实现 XSS 绕过。

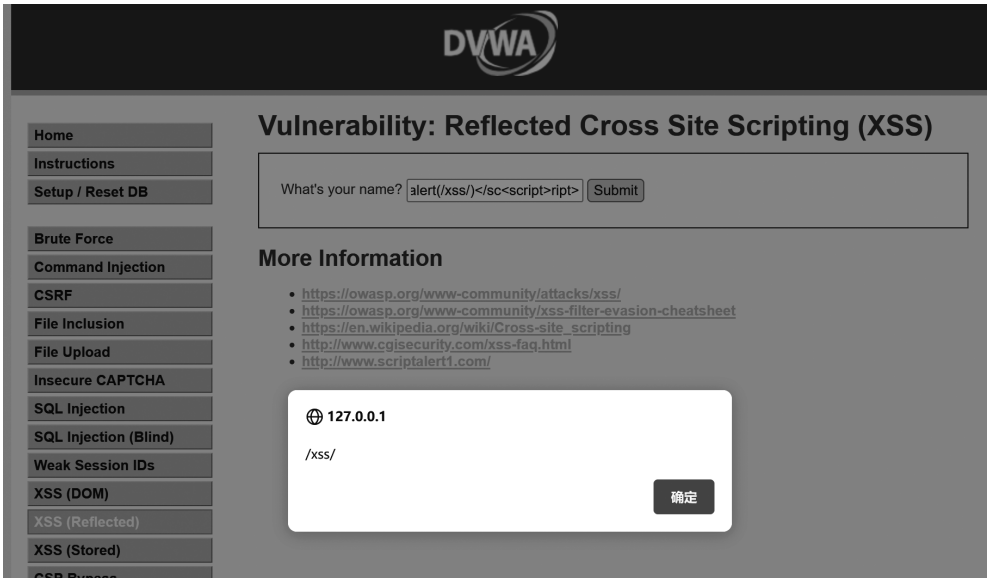

图 5-9 测试反射型 XSS 返回的弹窗

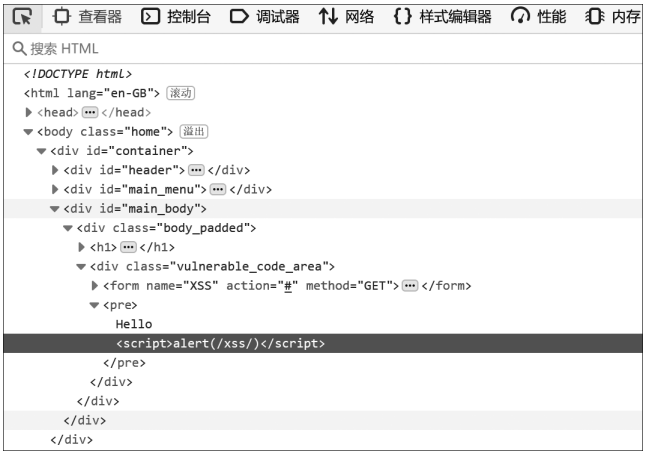

图 5-10 将过滤后的测试脚本作为 HTML 元素解释运行

5.3.2 存储型 XSS 漏洞测试

继续基于 DVWA 靶场测试存储型 XSS,安全级别设置为 Low。分析服务器 PHP 源代码并分析 XSS 漏洞,构造存储型 XSS 漏洞测试用例。

1. 服务器端 PHP 代码

```
<?php
if( isset( $_POST[ 'btnSign' ] ) ) {
    //Get input
    $message = trim( $_POST[ 'mtxMessage' ] );
    $name = trim( $_POST[ 'txtName' ] );
    //Sanitize message input
```