# 存储型XSS

存储型XSS也叫持久型XSS，从名字就知道特征是攻击代码会被存储在数据库等存储介质中，因此功效持久。攻击者可以把payload放在网站留言板、评论等位置，待用户访问网站并有匹配payload的行为时，即可触发攻击。

存储型XSS和反射型XSS本质都是一样的，只不过反射型XSS的payload的行动路径是：受害者浏览器--服务器--受害者浏览器，而存储型XSS的payload的行动路径是：浏览器--服务器--存储介质--服务器--受害者浏览器。

## LOW

### 通关步骤

1、试一下最简单的payload：<script>alert(1)</script>

这里我是写在message输入框里面的





name输入框是限制长度的可以在name输入框右键->查看元素（chrome浏览器里面是右键->检查），或者F12打开之后在elements里面自己找找。

可以看到name输入框maxlength是10，我们直接把maxlength的值改成100



可以正常输入了



弹窗成功



## 源码分析

<?php

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

    // Sanitize message input
    $message = stripslashes( $message );
    $message = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $message ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));

    // Sanitize name input
    $name = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $name ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"],  $query ) or die( '<pre>' . ((is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );

    //mysql\_close();
}

?>

表单提交后name和message中的内容首先被trim函数移除左右两边的字符，再被stripslashes()函数删除反斜杠（\），然后经过mysqli\_real\_escape\_string() 函数的处理，转义了特殊字符（包括NUL（ASCII 0）、\n、\r、\、'、" 和 Control-Z），然后直接代入mysqli\_query()函数来执行INSERT INTO的SQL语句。

完全没有对XSS的防护，另外对SQL注入的防护也不彻底。

## MEDIUM

### 通关步骤

1、这关<script>alert(1)</script>不好使了，显示的都只剩下alert(1)了





既然是删除式过滤，常规思路可以尝试以下大小写绕过，双写绕过之类的，不行的话试试没有script的payload。

这次我打算试试双写绕过。

考虑到删得连尖括号都不剩了，试试paylaod：<scrip<script>t>alert(1)</scrip<script>t>

成功弹框





因为写在message无效，所以注入点在name中

### 源码分析

<?php

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

    // Sanitize message input
    $message = strip\_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $message ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = str\_replace( '<script>', '', $name );
    $name = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $name ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"],  $query ) or die( '<pre>' . ((is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );

    //mysql\_close();
}

?>

比起LOW等级的代码，MEDIUM等级主要是有这三个高亮的地方的修改：

（1）对message的内容中的预定义字符之前添加反斜杠(addslashes函数)。预定义字符包括 '、"、\、NULL（其实感觉这步挺多余的，因为默认地，PHP 对所有的 GET、POST 和 COOKIE 数据自动运行 addslashes()，所以不应对已转义过的字符串使用 addslashes()，会导致双层转义，详细可见https://www.w3school.com.cn/php/func\_string\_addslashes.asp）

然后用strip\_tags() 函数剥去字符串中的 HTML、XML 以及 PHP 的标签。

（2）将message中的预定义字符转换为html实体（htmlspecialchars函数）。预定义的字符包含&、<、>、'、"

（3）将name中的<script>删除（使用str\_replace函数进行字符串替换，由于该函数是区分参数大小写的，所以也可以采用大写绕过）

本靶场存储型XSS注入的输出位置在html标签中间，因此由于message那里将包括<和>的预定义字符转换成了html实体，所以message处无法注入。

而name处仅仅是对<script>进行了删除，并且该删除操作采用的还是区分参数大小写的函数，而且没有循环删除。

因此name这里至少可以用三种方法绕过：（1）双写<script>绕过，如本文上述示例（2）大写绕过，比如<sCript>（3）换成不带<script>标签的payload

## HIGH

### 通关步骤

1、先用<script>alert(1)</script>试一下，name和message同时试一试

从结果来看，这个name字段的奇葩结果让我想起了反射型XSS的HIGH等级，那边是用正则表达式<(.\*)s(.\*)c(.\*)r(.\*)i(.\*)p(.\*)t来过滤的



HTML 的 **< img >** 标签定义 HTML 页面中的图像，该标签支持 **onerror 事件**，在装载文档或图像的过程中如果发生了错误就会触发。使用这些内容构造出 payload 如下，因为我们没有图片可供载入，因此会出错从而触发 onerror 事件。

<img src = 1 onerror = alert('xss')>



### 源码分析

<?php

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

    // Sanitize message input
    $message = strip\_tags( addslashes( $message ) );
    $message = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $message ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = preg\_replace( '/<(.\*)s(.\*)c(.\*)r(.\*)i(.\*)p(.\*)t/i', '', $name );
    $name = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $name ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));

    // Update database
    $query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
    $result = mysqli\_query($GLOBALS["\_\_\_mysqli\_ston"],  $query ) or die( '<pre>' . ((is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_error($GLOBALS["\_\_\_mysqli\_ston"]) : (($\_\_\_mysqli\_res = mysqli\_connect\_error()) ? $\_\_\_mysqli\_res : false)) . '</pre>' );

    //mysql\_close();
}

?>

源码如下，**preg\_replace()** 函数执行一个正则表达式的搜索和替换，“\*” 代表一个或多个任意字符，“i” 代表不区分大小写。也就是说 name 参数 “< script >” 标签在这里被完全过滤了，但是我们可以通过其他的标签例如 img、body 等标签的事件或者iframe 等标签的 src 注入 JS 攻击脚本。

## IMPOSSIBLE

### 源码分析

<?php

if( isset( $\_POST[ 'btnSign' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $\_REQUEST[ 'user\_token' ], $\_SESSION[ 'session\_token' ], 'index.php' );

    // Get input
    $message = trim( $\_POST[ 'mtxMessage' ] );
    $name    = trim( $\_POST[ 'txtName' ] );

    // Sanitize message input
    $message = stripslashes( $message );
    $message = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $message ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));
    $message = htmlspecialchars( $message );

    // Sanitize name input
    $name = stripslashes( $name );
    $name = ((isset($GLOBALS["\_\_\_mysqli\_ston"]) && is\_object($GLOBALS["\_\_\_mysqli\_ston"])) ? mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $name ) : ((trigger\_error("[MySQLConverterToo] Fix the mysql\_escape\_string() call! This code does not work.", E\_USER\_ERROR)) ? "" : ""));
    $name = htmlspecialchars( $name );

    // Update database
    $data = $db->prepare( 'INSERT INTO guestbook ( comment, name ) VALUES ( :message, :name );' );
    $data->bindParam( ':message', $message, PDO::PARAM\_STR );
    $data->bindParam( ':name', $name, PDO::PARAM\_STR );
    $data->execute();
}

// Generate Anti-CSRF token
generateSessionToken();

?>

源码在 high 级别的基础上对 name 参数也进行了更严格的过滤，导致 name 参数也无法进行 JS 脚本注入。同时加入 Anti-CSRF token 防护 CSRF 攻击，进一步提高安全性。

# 总结与防御

**跨站脚本 (Cross-Site Scripting)** 是一种针对 Web 程序的代码注入型漏洞攻击，它允许攻击者将恶意脚本注入网页，使得其他用户浏览网页时收到影响。所谓**存储型 XSS** 又称为持久型 XSS，攻击脚本会被永久保存在目标服务器的数据库或文件中，具有更高的隐蔽性。
存储型 XSS 的攻击常见于论坛、博客或留言等需要提交文本的页面，攻击者将攻击脚本和文本一起注入。当文本被服务器存储下来时，恶意脚本也会被永久存放与服务器的数据库或文件中。当其他用户访问这个含有恶意脚本的页面时，恶意脚本会在用户的浏览器中执行。XSS 漏洞的修复方式有以下 2 种：

1. **过滤输入的字符**，例如 “ ' ”，“ " ”，“<”，“>” 等非法字符；
2. 对输出到页面的数据进行**编码**。