### 反射型 XSS 攻击实战

#### 1. Low 级别反射型 XSS 攻击实战

步骤 1:安全级别设置为 Low,点击 XSS(Reflected)按钮,进入反射型 XSS 攻击模块,如图 1.1。



图 1.1

步骤 2:尝试提交弹窗脚本 <script>alert(document.cookie)</script>,输出用户 cookie,可以直接成功,说明 Low 级别未做任何防护措施,如图 1.2。

192.168.119.200/dvwa/vulnerabilities/xss\_r/?name=<script>alert(document.cookie)<%2F

fore you can access the Internet.

security=low; PHPSESSID=0niq9qbe4kp4qhimt9cv7g1lm6

图 1.2

## 2. Medium 级别反射型 XSS 攻击实战

步骤 1:安全级别设置为 Medium,再次尝试直接提交输出 cookie 脚本,发现把脚本内容直接显示出来了,说明对敏感的 JS 脚本做了过滤或转义,如图 2.1。

# Vulnerability: Reflected Cross Site S What's your name? Submit

Hello alert(document.cookie)

图 2.1

步骤 2: 查看页面源码,发现如下代码 \$name = str\_replace('<script>','', \$\_GET['name']),使用 str\_replace 函数把提交内容中的 <script> 替换为了空值,如图 2.2。

图 2.2

步骤 3: 考虑到 PHP 严格区分大小写字母,该替换函数只匹配的小写的 script,并没有匹配大写字母,尝试把 script 全部换成大写,提交 <SCRIPT>alert(document.cookie)</SCRIPT>,发现可以成功弹窗,如图 2.3。

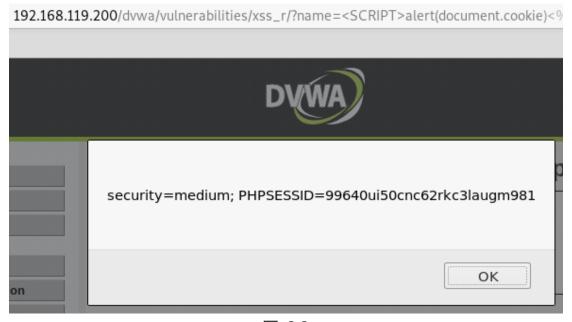


图 2.3

步骤 4: 该替换函数是对整个 <script> 字符做替换,而且只替换了一次,并没有做递归检查,尝试在 <script> 中再嵌套一个 <script>,提交 <scr<script>ipt>alert(document.cookie)</script>,也可以成功,如图 2.4。

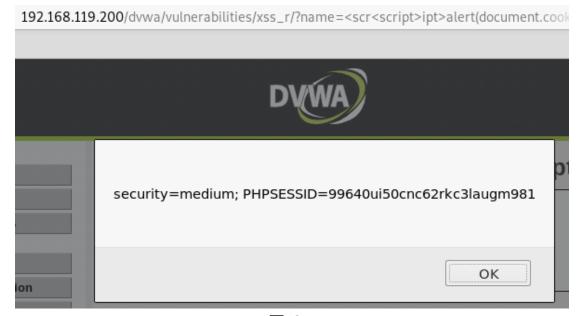


图 2.4

### 3. High 级别反射型 XSS 攻击实战

步骤 1: 尝试前面的攻击方法,发现在 High 级别下都无法成功,查看页面源码,发现如下代码 \$name = preg\_replace('/<(.\*)s(.\*)c(.\*)r(.\*)i(.\*)p(.\*)t/i','',\$\_GET['name']),如图 3.1。preg\_replace 函数可以调用正则表达式。我们发现该替换函数使用正则表达式进行了 script 的逐字检查,并通过 /i 来不区分大小写,所在造成之前的方法都不管用。



图 3.1

步骤 2: JS 脚本不仅仅可以在 <script> 标签中使用,通过 <img> 标签中onerror 行为也可以调用 JS 脚本。提交 <img src=1

### onerror=alert(document.cookie)>, 成功弹窗, 如图 3.2。

192.168.119.200/dvwa/vulnerabilities/xss\_r/?name=<img+src%3D1+onerror%3Dalert(d

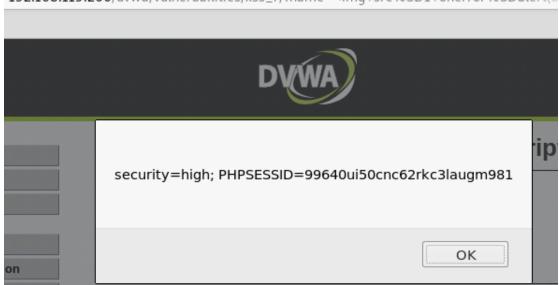


图 3.2