
# 浅谈移动养殖系统和IOS安全

安天移动安全研发中心

# 提纲

- 一：
- 威胁与风险
- 解决方案
- 系统介绍
- 技术支持

- 二：
- IOS安全防范

一：

# 前言 Introduction

Question：

动态养殖是什么？

为什么有这个需求？

带来的效应是什么？

# 威胁与风险

- 恶意样本数量增速
- 恶意样本种类繁多
- 当前的检测方式
- 恶意代码的检测问题

# 恶意样本数量增速



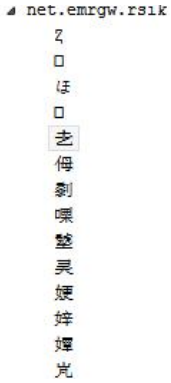

| | |
|---|---|
| 2015-11-24 | 5789431<br>投放总数：**73893**<br>进入筛分的新样本数：**53460**<br>投放筛分丢失的样本数：**0** |

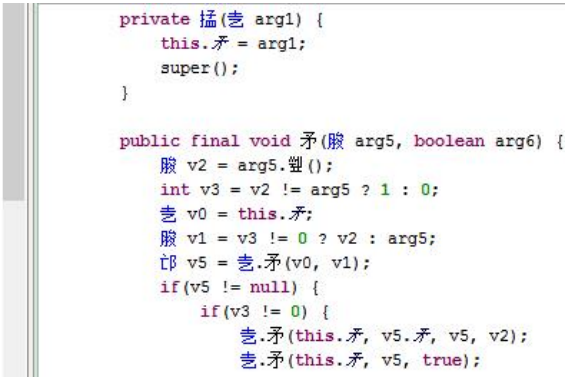
2700777
休眠中入：**17155**
休眠中出：**88**

**Question：增速之快怎么解决？数量之多呢？**

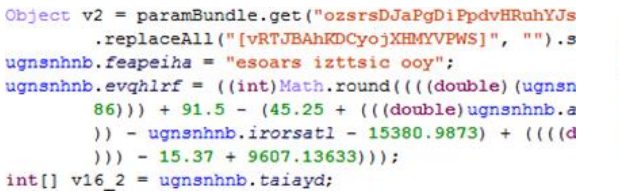
# 恶意代码种类繁多

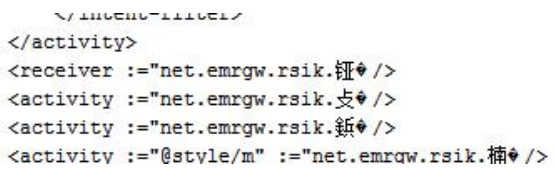








**Question：恶意代码种类的增多，人工如何面对？**
**例：so核心功能隐藏、混淆、防反编译、自校验等加大静态分析成本。**

# 目前检测方法的缺陷



静态
+
动态

静态

细粒度可疑函数分析
--分析耗时

病毒码针对性提特征

不能自动检测新型恶意软件

动态

分析恶意代码执行后行为

--检测能力更强

计算复杂消耗系统资源

虚拟环境无法实时与真实环境保持一致

# 恶意代码检测的问题




1. Dex、no Certificate、games无法检测
2. 动态分析技术无法够根据恶意代码行为需要，动态模拟检测环境以提升检测能力---例：模拟填写
3. 沙箱技术容易被恶意代码检测并逃逸check getDeviceId
4. 原生ROM镜像分析周期长

# 解决方案

- 需求
- 构思图
- 关键技术

# 解决方案----需求




=

# 解决方案----构思图



**•定制化监控参数设置**

- 自动化流程操作
- 定制化配置调节

**•丰富样本行为触发分析能力**

- 动态模拟满足系统核心资源
- 全面行为触发&反逃逸能力升级

## 动态养殖分析系统

- 基于Java堆栈原理，实现敏感代码调用追踪

**•恶意行为代码级函数调用追踪**

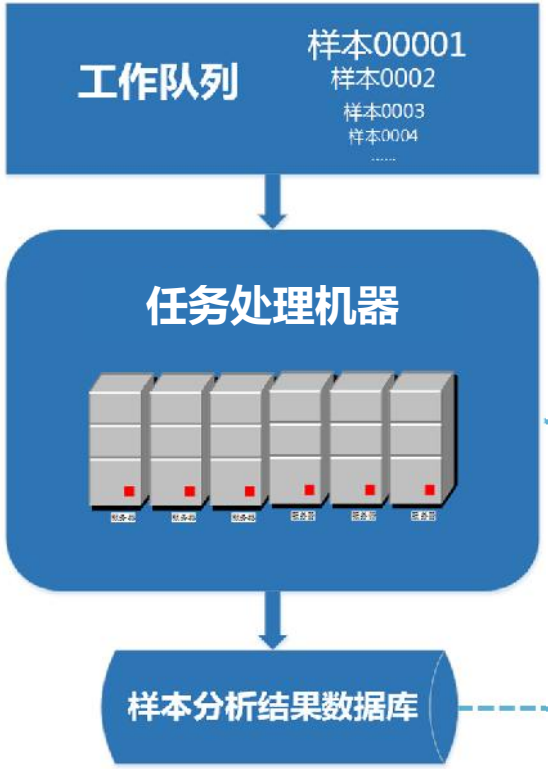
- 以时间为序进行行为触发展示
- 以危险性对行为分级
- 四种标准格式输出

**•详细结果展示和标准格式输出**

# 关键技术----自动化流程



## 1 自动化流程操作




任务流水线：C/S模式
满足一定条件时，自动投放样本工作队列

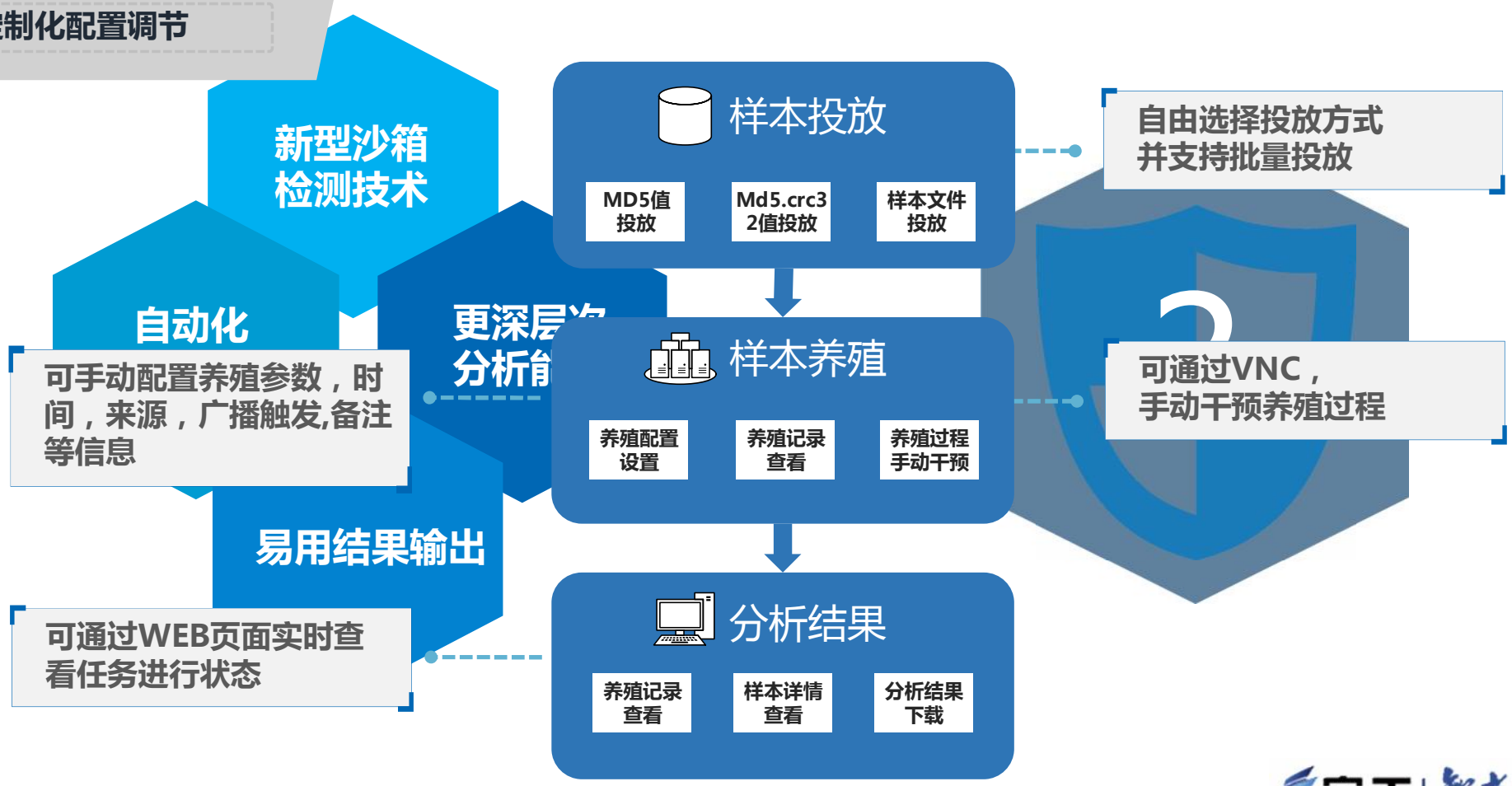
通过虚拟机批量部署，无需过多操作

养殖结果由流水线脚本自动化回收入库

# 关键技术----定制化监控参数设置



2 定制化配置调节



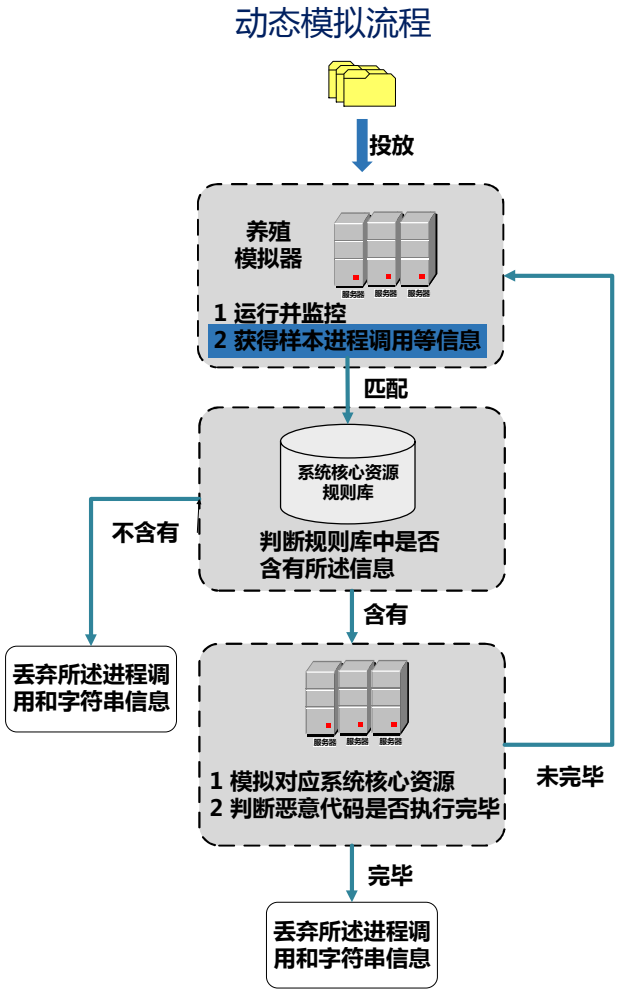
# 关键技术----丰富样本行为触发分析能力



## 1 ：动态模拟满足系统核心资源

1

当恶意代码未能执行完毕时，能够根据恶意代码行为需要，动态模拟满足系统核心资源，减少资源消耗，提升检测效率。



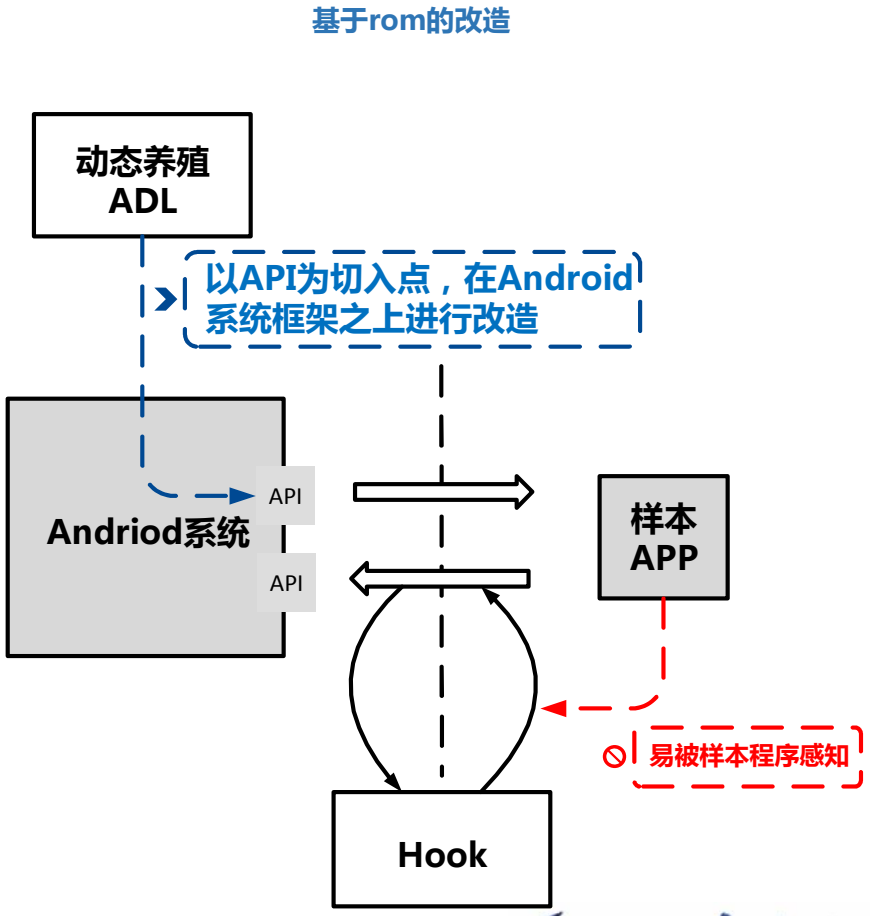
# 关键技术----ROM改造



2 ：行为触发&针对反逃逸

2

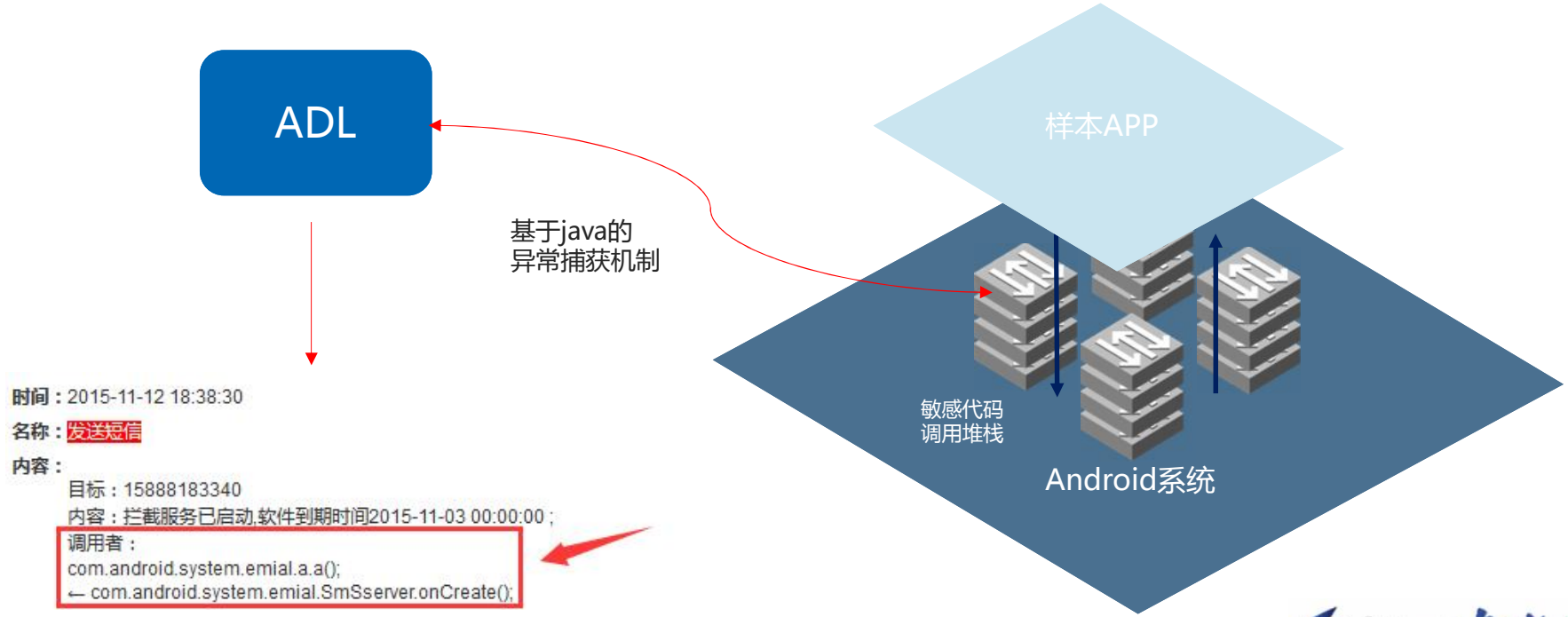
行为捕获基于rom的改造，对于应用是透明不可见，提升反逃逸能力



# 关键技术----恶意行为代码级函数调用追踪



1 对于样本行为有详细函数调用信息，便于进行代码级追踪分析。



# 关键技术----详细结果展示和标准格式输出



1 以时间为序的详细结果展示，以及行为捕获日志行形成标准化行为文档进行输出




以时间为序，对样本行为进行跟踪记录并展示

# 技术细节

- 定制ROM
- 定制化监控参数设置

# 技术细节----定制ROM



1 **技术细节：**

1：修改packages/apps/……

2：the key of function打印log

---http post

---SMS_send

---Device_admin

3：修改packages/apps/…… imsi&imei对抗 check monitor

# 技术细节----定制ROM



2 **技术细节：**

4： add TimerTask 定时发送短信

5： Analog Click

---应对emial

---行为触发

6：Ubuntu编译源码

# 技术细节----定制化监控参数设置



## 3 参数特点：

Application 层函数节点

**例：sendDataMessage 简化SMS_SEND**

| 类型 | 类型名称 | 类型说明 | 内容说明 | | 类型 | 类型名称 | 类型说明 | 内容说明 |
|---|---|---|---|---|---|---|---|---|
| 短 信 | SMSSF | 往目标号码短信发送 | 目标号码地址;;发送内容 | | 安装卸载 | INSTALLPKG | 安装应用程序apk | ADB/NOR;pkg Name |
| | SMSINTERCEPT | 拦截系统短信程序接收短信，收件箱中查看不到短信 | 拦截系统短信程序接收短信，收件箱中查看不到短信 | | | DELETEPKG | 卸载应用程序 | packageName |
| 底层操作 | EXEC/FEXEC | 执行Linux底层命令，如su、mount等命令 | 执行命令 命令参数 | | 录音 | STARTRECSND | 开始录音 | API调用记录 |
| 网络连接相关 | URLCREATE | 包含url 的URL对象创建 | url | | 拍照 | TACKEPIC | 开始拍照 | PREV:OPEN/CLOSE（是否开启预览） |
| | HTTPPOST | Http Post操作的URL | url | | 外部加载 | 外部加载 | 加载dex文件 | path |

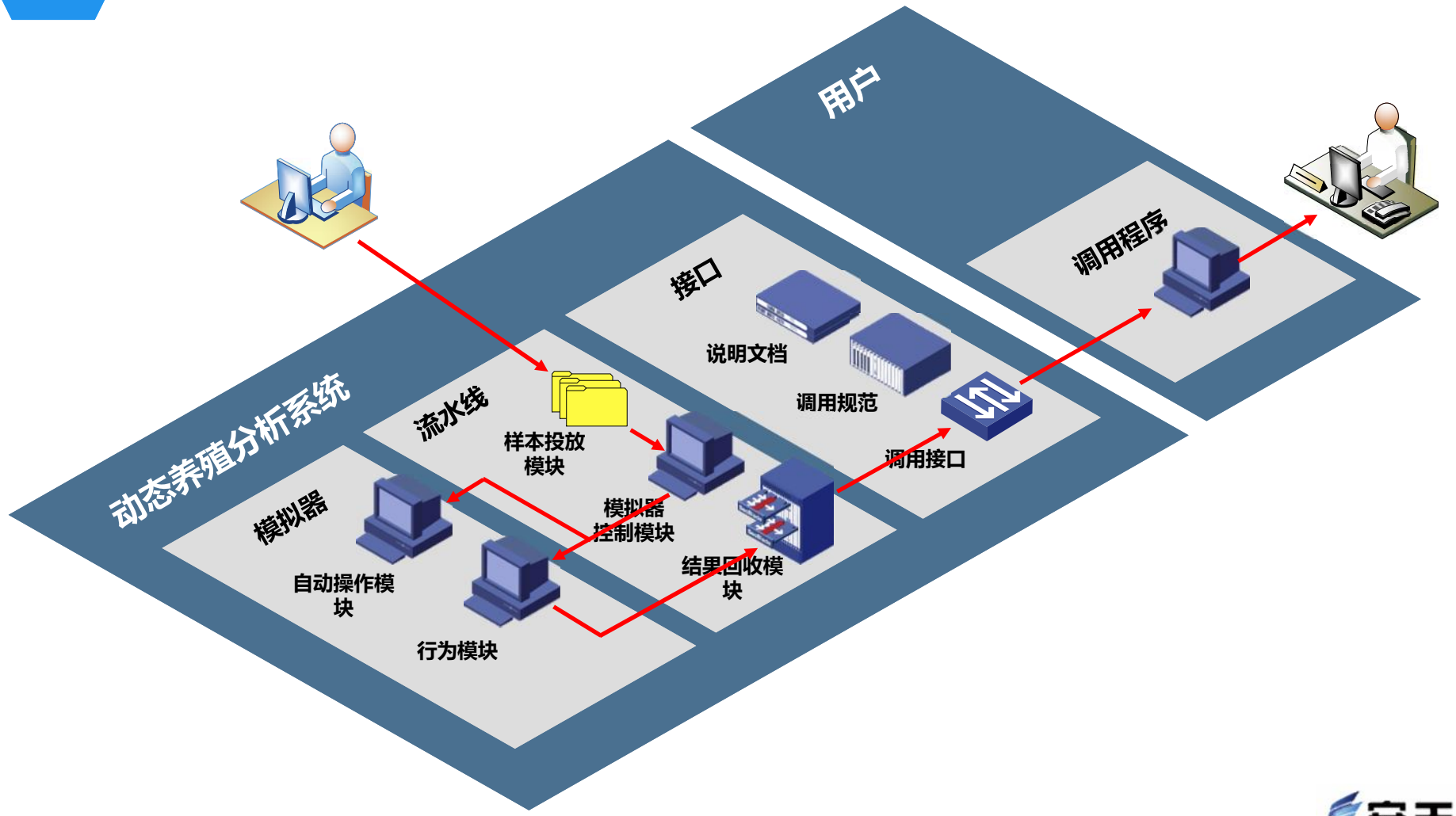
| | |
|---|---|
| android.intent.action.SIG_STR | 手机信号强度 |
| android.intent.action.BATTERY_CHANGED | 电量改变 |
| android.intent.action.BOOT_COMPLETED | 启动完成 |
| android.intent.action.TIME_TICK | 时钟广播 |
| android.intent.action.SCREEN_ON | 屏幕解锁 |
| android.intent.action.SCREEN_OFF | 锁屏广播 |
| aandroid.intent.action.MEDIA_MOUNTED | SDCard挂载 |
| android.intent.action.UMS_CONNECTED | USB进入U盘模式 |
| android.intent.action.CALL | 拨打电话 |
| android.intent.action.DIAL | 拨打电话 |
| android.intent.action.CFF | 语音电话的呼叫转移状态已经改变 |
| android.intent.action.CONFIGURATION_CHANGED | 设备的配置信息 |
| android.intent.action.DATA_ACTIVITY | 电话的数据活动(data activity)状态已经改变 |
| android.intent.action.DATA_STATE | 电话的数据连接状态已经改变 |
| android.intent.action.DATE_CHANGED | 日期被改变 |
| android.server.checkin.FOTA_CANCEL | 取消所有被挂起的 (pending) 更新下载 |
| android.server.checkin.FOTA_INSTALL | 更新已经被确认，马上就要开始安装 |
| android.server.checkin.FOTA_READY | 更新已经被下载，可以开始安装 |
| android.server.checkin.FOTA_RESTART | 恢复已经停止的更新下载 |
| android.server.checkin.FOTA_UPDATE | 通过 OTA 下载并安装操作系统更新 |
| android.intent.action.PACKAGE_ADDED | 设备上新安装了一个应用程序包 |
| android.intent.action.PACKAGE_REMOVED | 设备上删除了一个应用程序包 |

# 系统介绍

- 系统体系结构
- 成熟案例应用
- 产品实际案例

# 安天动态养殖分析系统----系统体系架构

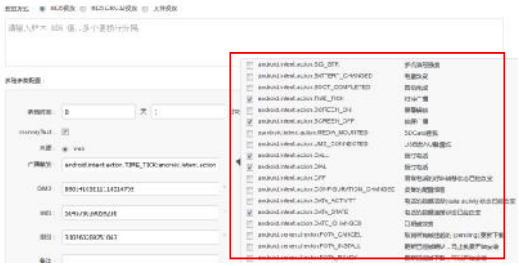




# 安天动态养殖分析系统----成熟案例应用



## 1 任务分发



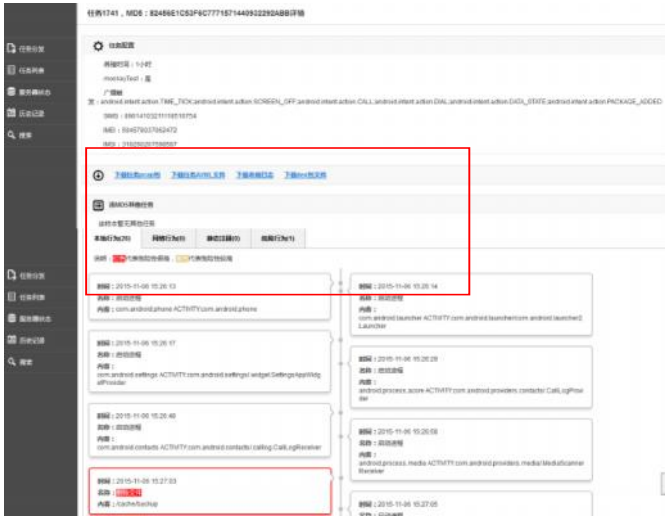
## 2 任务列表

## 3 服务器状态

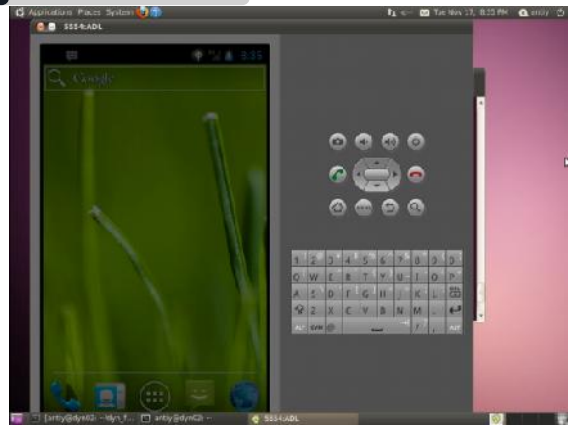
## 4 历史记录

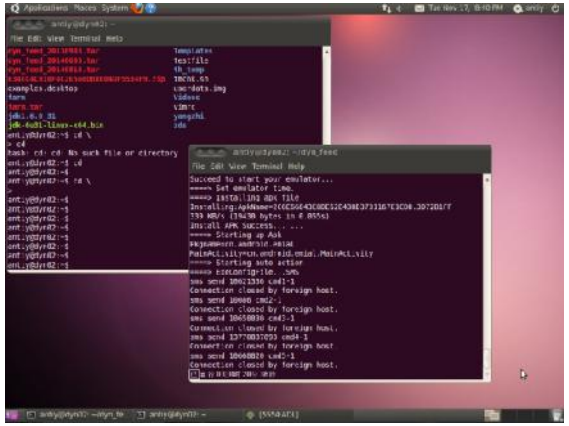
## 5 任务分发

## 6 样本养殖详情



## 7 查看VNC





# 安天动态养殖分析系统----产品实际案例



MD5：70E62332ECD56EF3C73F316F944DAA3D

解密获取手机号码

**时间：**2014-07-12 02:00:39**名称：**解密

**内容：**输入：2e3def3af08b4c57151e2ace7606400e(H);
输出：1851069843 ;
调用者：
com.example.test.DesUtils.decrypt(DesUtils.java);
← com.example.test.DesUtils.decrypt(DesUtils.java);
← com.example.test.u.getData(u.java);
← com.example.test.n.getSendTo(n.java);
← com.example.test.n.chkIsFirstRun(n.java);
← com.example.test.u.chkFirstRun(u.java);
← com.example.test.la.onCreate(la.java);

解密获取数据

**时间：**2014-07-12 02:00:39**名称：**解密

**内容：**输入：af707591cd7a19633bc18e1940d7e59b04aa14b6d827d392(H);
输出：2014-05-01 11:30 ;
调用者：
com.example.test.DesUtils.decrypt(DesUtils.java);
← com.example.test.DesUtils.decrypt(DesUtils.java);
← com.example.test.u.getData(u.java);
← com.example.test.n.getSendTo(n.java);
← com.example.test.n.chkIsFirstRun(n.java);
← com.example.test.u.chkFirstRun(u.java);
← com.example.test.la.onCreate(la.java);

激活设备管理器

**时间：**2014-07-12 02:00:40**名称：**调用类方法

**内容：**
android.telephony.SmsManager.sendMultipartTextMessage("1851069843","null","[安装成功啦]","null","null");
← ;
调用者：
com.example.test.Tx.tx(Tx.java);
← com.example.test.u.sendMsg(u.java);
← com.example.test.n.chkIsFirstRun(n.java);
← com.example.test.u.chkFirstRun(u.java);
← com.example.test.la.onCreate(la.java);

**时间：**2014-07-12 02:00:40**名称：**启动活动

**内容：**
com.android.settings.DeviceAdminAdd 行为:android.app.action.ADD_DEVICE_ADMIN ;
调用者：
com.example.test.la.onCreate(la.java);

**时间：**2014-07-12 02:00:40**名称：**发送短信

**内容：**
目标：1851069843
内容：安装成功啦 ;
调用者：
java.lang.reflect.Method.invokeNative(Native Method);
← java.lang.reflect.Method.invoke(Method.java:530);
← com.example.test.Tx.tx(Tx.java);
← com.example.test.u.sendMsg(u.java);
← com.example.test.n.chkIsFirstRun(n.java);
← com.example.test.u.chkFirstRun(u.java);
← com.example.test.la.onCreate(la.java);

# 数据挖掘

## 对数据进行回溯

污点数据：

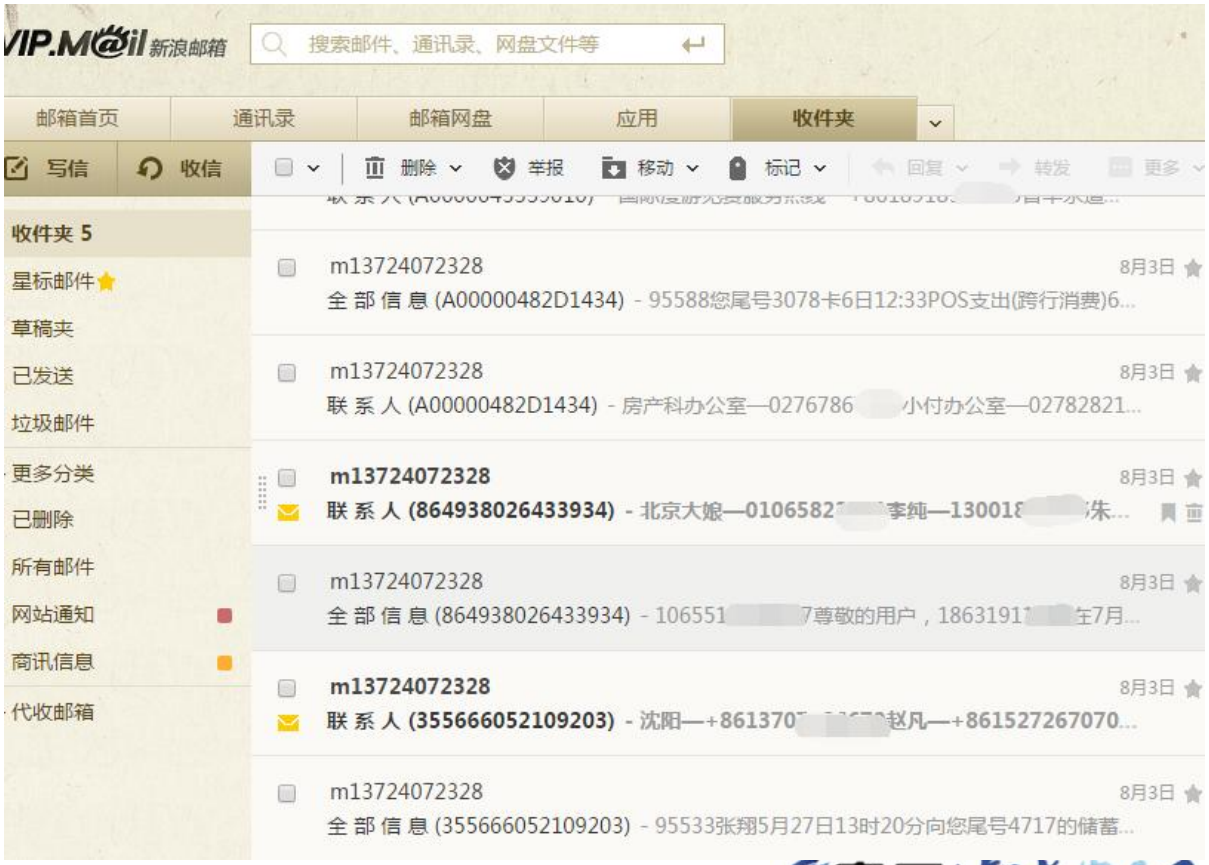
时间：2014-09-05 16:10:51名称：发送短信

内容：

目标：1368584[redacted]
内容：提示：设备已安装，等待激活；
调用者：
me.rt6t54rjhf.try5tfhfg.rt4dfhfgj.tool.MeTianxiaOuChanUtil.se(MeTianxiaOuChanUtil.java:56);
← me.rt6t54rjhf.try5tfhfg.rt4dfhfgj.activity.MeTianxiaOuChanActivity.onCreate(MeTianxiaOuChanActivity.java:21);

定位：


污点数据：



# 二：

## 前言 Introduction

Question：

IOS非常安全？

没有越狱呢？

隐私泄露的代价？

# IOS安全事件----未越狱设备

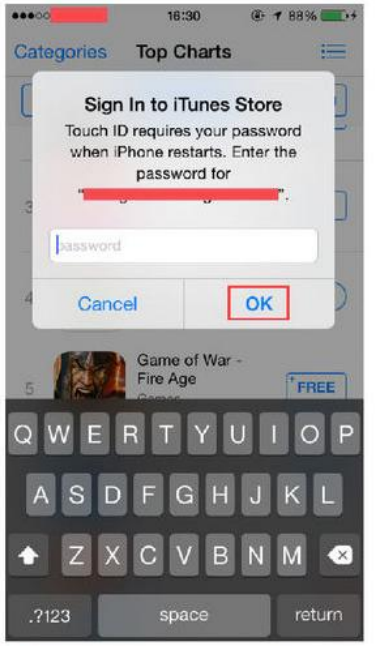


事件1： Hacking Team泄露样本无需越狱即可监控iOS用户
--2015年7月

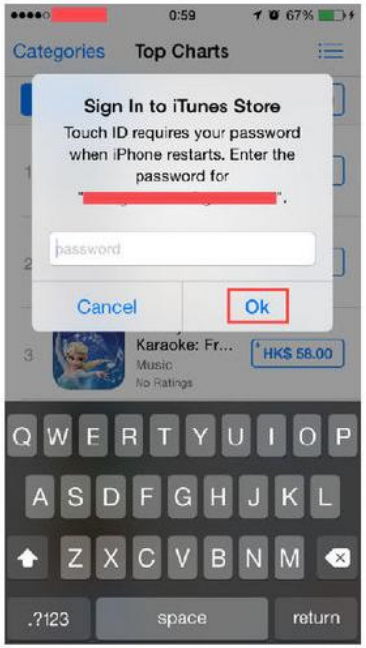



事件2： CVE-2015-5838 触发沙箱逃逸
-- 2015年9月

Ios8.1未越狱设备，伪造app弹出虚假登录框盗取Apple ID




原版的AppStore登录窗口




伪造的AppStore登录窗口

# IOS安全事件----越狱设备



**事件3： KeyRaider盗取超过22.5万苹果账户**
**--2015年9月**


共享Appleid免费购买应用

```
if ( std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::find(
       &v16,
       "ituneesstored",
       0,
       12) != -1 )
{
  v18 = 3;
  std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::__init(
    &v15,
    "/System/Library/Frameworks/Security.framework/Security",
    54);
  v18 = 4;
  std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::__init(&v14, "SSLWrite", 8);
  v18 = 5;
  v11 = getAdress(&v15, &v14);
  v18 = 6;
  std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::~basic_string(&v14);
  v18 = 7;
  std::__1::basic_string<char,std::__1::char_traits<char>,std::__1::allocator<char>>::~basic_string(&v15);
  if ( v11 )
  {
    v18 = 8;
    MSHookFunction(v11, replace_SSLWrite, &original_SSLWrite);
  }
```

获取appleid账号密码

```
_query_label = objc_msgSend(_query, "objectForKey:", CFSTR("labl"));
if ( _query_label )
{
  v69 = -1;
  NSLog(CFSTR("==labl:%@"));
  v69 = -1;
  if ( !objc_msgSend(_query_label, "compare:", CFSTR("com.apple.lockdown.identity.activation")) )
    *(_DWORD *)_result = 0;
```

# IOS安全实际案例



**国内丢失iphone你可能会出现以下这种情况：**
**----销赃的利益链**
**---- 2015年12月**

Iphone丢失 → 补办卡/登录邮箱




一个月内成功骗到一万三千多个**iCloud**账号和密码

追溯 ← 流入二手市场销售 ← 手机解除Appleid绑定


# IOS结语



- 1：Appleid账户密码很重要，决定了你手机何时报废、包括你的ipad、macbook
- 2：只从AppStore上下载应用，勿信任任何企业证书
- 3：勿越狱IOS设备、酌情更新漏洞补丁

# 谢谢大家

THANK YOU FOR YOUR ATTENTION

www.antiy.com

安天 | 智者安天下