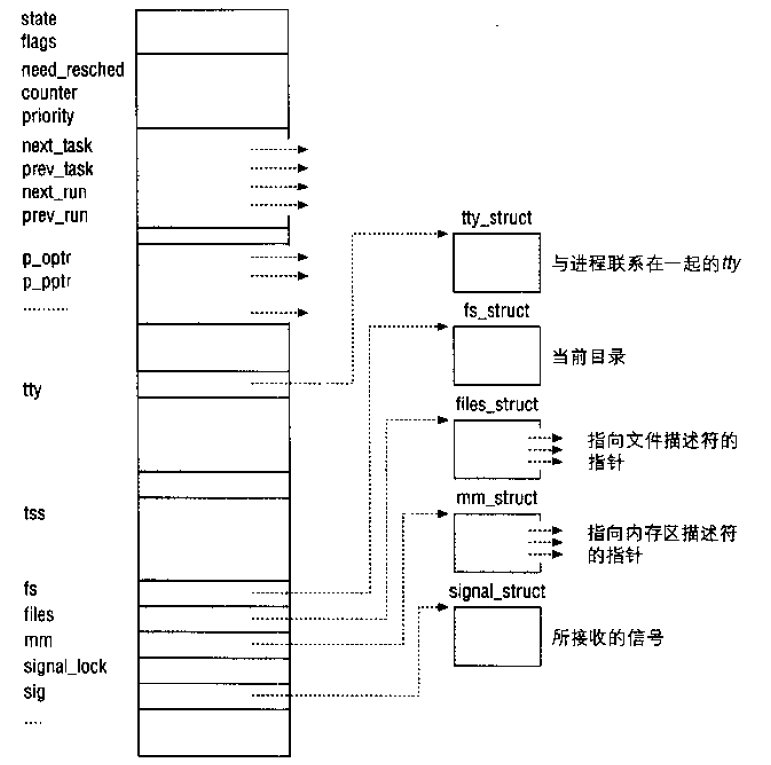
基于Android内存镜像的恶意软件取证研究

1. 绪论
   1. 研究背景与意义
   2. 国内外研究现状
      1. 恶意软件检测研究
      2. 内存取证研究
   3. 论文主要工作
   4. 论文的结构安排
2. 相关知识介绍
   1. Android综述
      1. Android体系结构
      2. Android应用程序概述
   2. Dalvik虚拟机和dex文件
      1. Dalvik虚拟机
      2. dex文件结构
   3. Android系统面临的安全威胁
      1. APK重打包技术
      2. 动态加载技术
      3. 运行时自修改技术
   4. 内存取证相关介绍
      1. 内存镜像获取方法
      2. Linux内存管理方式
3. Android内存镜像分析
   1. Linux进程调度管理
      1. Linux调度器
      2. 相关数据结构
   2. 隐藏进程检测方法
      1. 隐藏进程常用方法
      2. 隐藏进程检测方法
   3. 可疑进程检测方法
      1. 恶意软件的一般操作
      2. 可疑进程检测方法
4. 基于概率神经网络的恶意软件检测
   1. APK反逆向分析
   2. 高危API提取
   3. Android恶意软件分类
5. 实验与结果分析
   1. 实验环境介绍
      1. 相关工具介绍
      2. 实验环境准备
   2. 内存镜像获取
      1. Android内存镜像获取
      2. 实验结果分析
   3. 隐藏进程检测
      1. 实验设计
      2. 结果分析
   4. 恶意软件检测
      1. 实验设计
      2. 结果分析
6. 总结与展望

常用进程隐藏方法：

1. 简单隐藏，替换系统调用等；
2. 文件隐藏，VFS隐藏/proc下面的进程文件系统；



进程描述符task\_struct

可以被调度的进程必须具有自己的进程描述符；

Pre\_task和next\_task共同组成了进程链表；

For\_each\_task的宏来进程扫描整个进程链表；

TASK\_RUNNING状态的进程链表：当内核CPU需要调度一个新的进程的时候，只考虑可运行进程（处在TASK\_RUNNING状态的进程）。如果扫描全部的进程列表是比较低效的，所有其具有一个可运行状态的进程链表，也叫运行队列(runqueue)，进程描述符中的next\_run和prev\_run域用来实现运行队列。

Init\_task进程描述符起秒表头的作用，nr\_running变量存放可运行队列中进程的总数；

进程切换：每个进程可以拥有属于自己的地址空间，然后所有的进程必须共享CPU寄存器，因此在调度进程的执行之前，需要确保每个寄存器装入了新进程的运行时的值。寄存器的一组数据叫做硬件上下文（hardware context），是可执行上下文的一个子集，存放在TSS段，对应进程描述符中的tss域。

Linux进程调度：

Linux2.4版本的进程调度是采用的O(n)调度器，是基于静态优先级来实现的，原理是其为系统内每个CPU维护一个全局运行队列，调度时每次找到该队列中优先级最高的那个进程并为其分配CPU。O(n)调度器的性能受全局运行队列的任务数量约束，就绪进程越多则查找最高优先级进程所花时间就越长，每个进程所需要分配的时间片也很难确定。此外，任意一个处理器调度都需要访问全局的运行队列，为了处理各处理器因访问队列而产生的竞争需要额外加锁，从而产生了新的性能瓶颈。为了改进性能，Linux2.6及以后的版本使用了O(1)调度器，其在进程调度上所需要的时间是恒定的，和就绪队列的任务数目没有关系。

可疑进程发现:系统调用、网络访问、文件和内存修改等；

检测.so和网络模型的都可以当作可疑行为，减少后序分析的进程的数量；因为一些简单的恶意软件在静态分析的时候就可以被发现，从而在受害者的手机里面发现的几率比较小，所有我们为了减少后续分析的内容，假设恶意软件进行了一些特殊的隐藏手段。而实现这些手段，则需要外部资源包的加载等信息；