用于自携装置安全之智能机虚拟化技术

介绍

根据 Wisegate 最新的 IT 战略研究报告[1],随着智能手机设备和移动应用和服务的快速增长,自带设备(BYOD- bring your own device)的安全性不仅已成为最大风险之一,它也变成了普通数据泄露和恶意软件,内部和外来威胁,和高级持续性威胁。此外,BYOD 和云安全已经被确定为最有影响力的趋势 IT 安全计划。一般 BYOD 安全所面临的挑战可以简单地说,许多移动应用程序都比较容易出现恶意软件影响的智能手机,但员工不知道的是,比如喜欢用智能手机访问企业网络。用户安装的各种应用程序和智能手机的潜在安全漏洞很快就会成为潜在的对恶意软件的网关渗透到公司的网络,数据库和其他系统。

通过安装安全监控到智能手机[2],MDM(移动设备管理)作为有效工具,已被广泛部署为来管理 BYOD 安全。但是,在智能手机中添加严格的安全控制代理,要在安全性和便利性之间进行强有力的权衡。一般情况下,员工更喜欢自由使用自己的智能手机,并且没有公司代理的任何干预或隐私防护,而公司有义务保护每一台设备。其结果是,一个安全的公司的智能手机现在往往只用于处理简单而琐碎的工作任务,同时员工还是会继续携带自己不不安全的设备到工作地点。

因此，现代 BYOD 安全的根本性挑战，不仅仅是安全管理能力，同时也是找到一个方式，让员工继续没有任何麻烦的使用 BYOD 智能手机，来有效和高效地工作。虚拟化技术被证明是解决这一挑战的聪明之举，因为它的虚拟机（VM）的完全隔离性质。因此，安全威胁是完全虚拟机内部隔离，即使操作系统被通过内核 rootkit 攻破。其结果是，一个单一的智能电话中运行多个独立的和虚拟移动操作系统，可以方便的提供企业级的安全和个人设备之间的最佳平衡，因为用户可以自由地操作他/她自己的智能手机，而潜在的安全损失被很好的隔离在别的 VM。

智能手机虚拟化的方式

始于 20 世纪 60 年代虚拟化技术指创造东西的虚拟版本的行为，比如计算机硬件，操作系统，存储设备和计算机网络。在过去的十年中，虚拟化技术已经取得成功的关键，以云计算或任何利用云计算的其他服务，如亚马逊和 Facebook 的进步。这是由于虚拟化的本质，更好的资源配置，整合，方便，经济，等。有了智能手机的硬件能力增强，以及智能手机使用率的快速增长，研究人员和公司一直在努力把虚拟化应用到智能电话的领域，解决各种问题[3][4]。领先的虚拟化解决方案提供商如 VMware 和思杰一直致力于迁移先进 VDI（虚拟桌面基础架构）到智能手持设备和执行面向企业 BYOD 安全的全面 MDM 解决方案。在另一方面，许多其他新兴创业公司，例如作为塞拉， Hypori，

Romotium 和 Cellrox 特别着眼于培养智能手机虚拟技术。

在本文中，两种类型的智能手机虚拟化，虚拟智能手机[6]和移动基础结构（VMI）[7]中，提出了建立一个单独或联合的新型 BYOD 安全解决方案。

Figure 1 Multiple virtual smartphones running on a single virtualized smartphone solving BYOD security management

这两种类型的智能手机虚拟化，一般都是根据，要求其客户端是否是瘦客户端或富含智能功能的智能手机。虚拟化的智能手机需要强大的手持式装置，它允许另一个 Android 操作系统正在对在虚拟机的形式的相同智能电话上运行（图 1）。在另一方面，VMI 可以让
们在远程服务器上运行 AndroidOSVM，通过网络流传输流回的 VM 智能手机屏。

虚拟化手机

由于虚拟化的智能手机配备有虚拟机管理程序，这是一个允许多个移动操作系统，能够同时在同一部智能手机上运行的技术，安全策略可以通过虚拟机管理程序，应用到每个访客移动操作系统，为每个虚拟手机，解决 BYOD 的关注点。例如，现在一个智能手机可以运行四个虚拟智能手机，即一个工作，安全，个人，一次性虚拟智能手机。如图 2 所示，一个工作的虚拟智能手机应严格锁定使用证明过程，如 BIOS，操作系统，应用程序，及其配置的二进制文件的验证。此外，一些企业安全策略可能会限制到只能观看，一些文件和不能被直接发送出去，在高度安全敏感公司，一个军用级认证，如 DISA 的安全技术实施指南（STIG）也可能需要被应用。

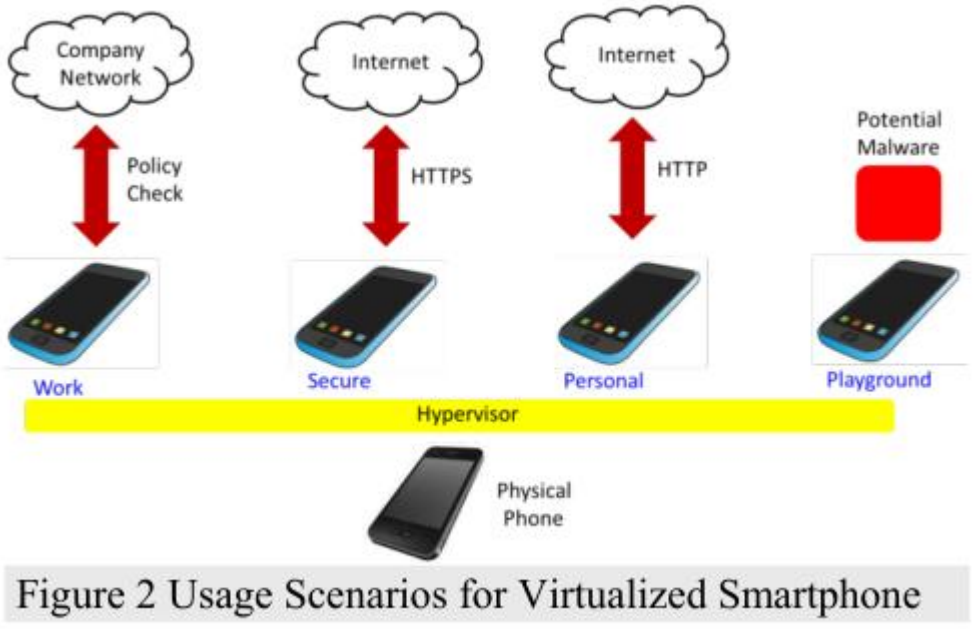

Figure 2 Usage Scenarios for Virtualized Smartphone

图 2 的虚拟化手机的使用场景

当敏感电子交易应用银行需要干净，安全的环境，安全的虚拟智能手机就可以应用。因此，基于白名单的安全保护可以应用，以确保没有未知的二进制被允许运行，并且不存在键盘记录器。对于个人虚拟智能手机，它只能使用黑名单防病毒解决方案来保护，用户可以安装并使用他们希望的任何类型的应用程序。最后，可以让用户创建一个虚拟的一次性智能手机作为“做你想做的”游乐场或应用测试沙盒。

提供一个成功的虚拟化的智能手机的主要技术挑战包括：

1） 智能手机上低开销的虚拟机管理程序-尽管今天的智能手机被配备有越来越多的 CPU 处理能力以及更大的内存（最大 4G），对有效地同时运行多个虚拟智能手机，仍然是一个大挑战。事实证明，智能手机上的一个低开销的管理程序，成为一个主要的竞争优势。有了这个低开销的虚拟机管理程序，就有可能 2.）在虚拟职能手机内部提供相同的 UI 流畅性 3.）并且可以在虚拟智能手机之间无缝的上下文相关切换。让智能手机虚拟化之后，如何执行安全策略到每个虚拟智能手机的出现将成为下一个重要任务。提供 4）给虚拟机的技术基于内省的白名单，5）仅显示文件系统：具有文件永远不会离开服务器和 6）恶意软件检测是安全的基本要求之一。对于一些总体思路恶意软件检测可能包括：

（1）来源跟踪：程序是否正确安装？

（2）安装检查：未正确安装的应用程序往往以不同的方式启动（例如在系统启动脚本中）。

（3）基于诱饵的检测：具有诱人力的文件名称会自动创建并放置在随机位置，不应被合法人员触及应用。

提供只显示文件系统服务，并确保文件永远不会离开服务器，APP 流的技术,如由 Agawi 提供的解决方案(获得由谷歌),VMFive,mNectar，App.io，AppSurfer 和 VOXEL 是正确答案。通过运行一个移动应用程序，而无需离开企业数据中心，这样数据能够安全地保持在企业内部 IT 系统里。

虚拟移动基础设施（VMI）

APP 流媒体可以被看作是一个虚拟移动基础设施（VMI）的简化版本只流单应用程序，而不是整个虚拟智能手机 VM 的活动。[8][9]对于 BYOD 安全，VMI 可能是最好的，跨平台和瘦客户机解决方案，这类似于桌面电脑的 VDI。

虚拟移动基础设施保持数据和应用在企业数据中心，因此没有数据泄露的危险。

图 3 呈现 VMI 的基本概念，它包括移动桌面流和局部传感器重定向两个概念。诸如 GPS，陀螺仪和多点触摸事件。在处理有穷举计算功耗方面，VMI 是特别有用。例如编辑和查看 CAD/CAM 的应用程序;网络大文件用例，视频和丰富的网络内容浏览;和高数据安全需求，如集中数据管理。最重要的是，VMI 的最好的部分是真正的“绝无设备丢失和被盗风险”。然而，接入带宽的变化和企业网络的限制仍然是更好的 VMI 用户体验的主要路障。此外，在开发商业级 VMI 时，富传感器装置，即时设备交互，也带来了显著的技术挑战解。

此外，在 VMI 服务器端，在 x86 服务器有效运行 Android，扮演另一个关键的作用，以及如何确保各种应用可以不加修改的适当运行，也可能是一个潜在的挑战。此外，客户端感测装置，如触摸传感器，陀螺仪传感器，GPS，摄像头，以及其他可能还需要配备到的 AndroidVM 使用各自的虚拟设备接口，这对设备仿真和即时结果流有进一步的困

难。

虚拟化手机

由于 Android 内核修改 Linux 内核，在 Linux 中现有的虚拟化解决方案可以方便地移植到 Android 环境。一个和 Android 内核的区别，主要在附加快速 IPC 机制，降低了应用/服务框架通讯的努力，以及应用保存的省电的细粒度控制的唤醒锁机制。

在本文中使用所提出的虚拟化解决方案是 KVM（内核基于虚拟机）内核模块和 QEMU（QuickEmulator）软件。该 KVM 依赖于英特尔 VMX 或 AMD 的 SVM 硬件来配置虚拟化的 CPU，内存虚拟化和控制从 VM 模式进入/退出/。该 KVM 提供用于用户级进程的 KVM API 创建 VM 和分配其相关联的 VCPU，存储器和在 VM 模式运行。QEMU 的进程调用相应的 API KVM 初始虚拟机，并且当虚拟机通过 I/O 设备端口来访问虚拟机时，来模拟它的 I/O 设备的活动。此外，QEMU 是还负责提供对虚拟机，网络端口转发到网络接入交付基础设施的功能，如网络地址转换（NAT)的将 VM 的服务访问点导出到公共网络和 VNC(虚拟网络计算）服务器提供了远程控制台访问 VM。

提出的 Android 虚拟化应用程序的软件架构如图所示图 4，它是类似于 Linux 的虚拟机管理程序软件：KVM 模块中运行内核模式下，QEMU 组件在运行用户模式和一个额外的虚拟化应用程序组件在

AndroidDavlik 之上运行 JVM。

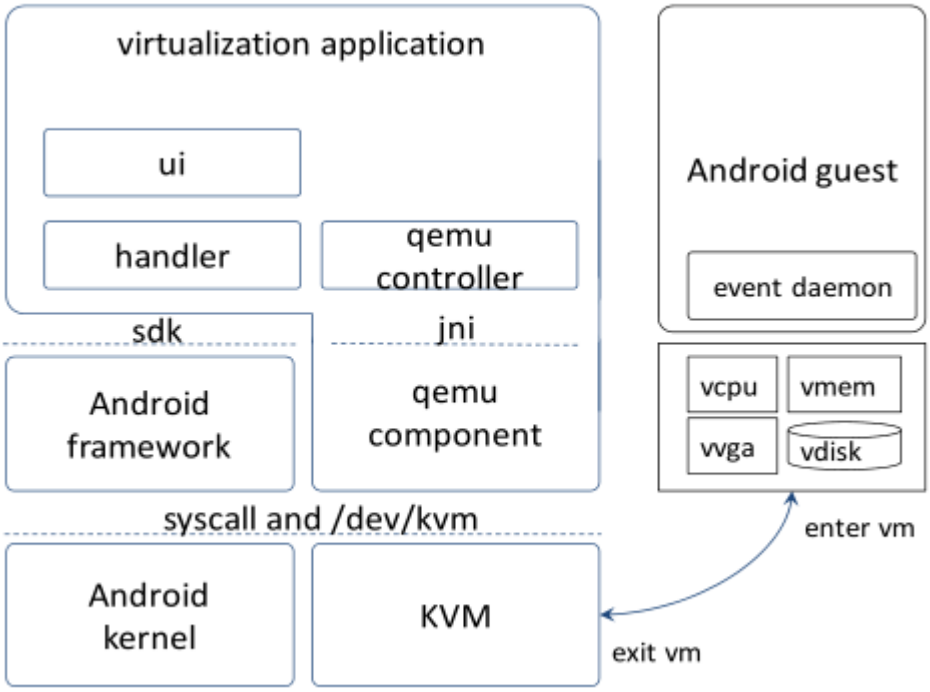

图 4，Android 虚拟化软件架构

由于 Android 框架是一个基于 Java 的执行环境，而 QEMU 取决于 C 库，它需要 JNI（JavaNativeInterface）在 QEMU 和 Android 应用程序之间进行通信。在图 4 中，Android 框架交互与 app 组件通过 AndroidSDK 中的接口，以及应用程序组件包含一个 QEMU 控制器，通过 JNI 接口调用 QEMU 组件的主要功能。虚拟机配置以方法参数的形式，传递给启动虚拟机的函数，包括虚拟 VCPU 的数量，内存的大小，磁盘映像文件的名称和网络设置。然后 QEMU 通过 KVM ioctl 命令将 KVM 内核模块调用到“/dev/kvm”用于创建虚拟机，然后设备调用 kvm_run 的 ioctl 命令进入 VM 执行模式。

虚拟智能手机 VM　APP

由于 Android 框架是用 Java 编写和执行在 JVM(Java 的虚拟机)中，Android 应用也通过 Java 字节码来访问框架的 API 和服务。虚拟化应用程序的上半部分也用 Java 编写的，包括主要有以下三个包：用户界面，处理器和 QEMU 控制器。用户接口包的主要目的是处理 UI 事件，刷新虚拟机的显示，当获取到点击事件时，调用其他软件包。QEMU 的控制器由用户接口包调用，有能力提供 API 来控制或查询 QEMU 状态，以及获取 VGA 帧缓冲和脏状态的指针。QEMU 的模块最终是由 QEMU 控制器通过 JNI 接口来执行，用来访问内存块或调用 C 代码中的函数。

处理器负责通过 SVMP 协议将主机中的事件转到 VM 的时间守护进程去。 [10][11][12]SVMP 协议从 SVMP 项目借用[4]，一个开源 VMI 项目，其中包含手机的各种事件的定义。例如：多点触摸事件，GPS 定位事件和电话事件旋转。这些活动将通过处理程序包接收并进行相应的转发到 Android 的虚拟机。相反地，Android 的虚拟机内部产生的事件也被事件守护进程传送到处理器包。例如，该 Android VM 中的通知消息都被封装成 SVMP 消息并传递给处理程序包以在主机的通知面板上显示。

显示优化

在 QEMU 模块内运行的 VM 输出显示内容到主机环境的模拟的 VGA 卡

上。用户可以使用远程显示协议或者 SDL 库来获得 VGA 帧缓冲器，在物理显示器设备上显示它。然而，这两种方法在 Android 手机上都不可行的，因为 Android 的环境不支持 SDL 库和远程显示协议的性能太慢由于不必要压缩和帧缓冲存储器复制操作。其结果是，为虚拟机显示一个较好的方法，提出了用下面的优化：

（1）直接访问驻留在虚拟中的帧缓冲内存机。

（2）使用主机手机的 3D 芯片加速显示速度。

为了直接访问帧缓冲，就必须寻找到了 QEMU 的 VGA 卡仿真代码并添加一个 JNI 函数 get_framebuffer（），返回 VGA 的 VRAM 帧缓冲区的指针。UI 组件显示刷新的代码，运行在 Java 字节代码层级，调用的 JNI 函数来获取帧缓冲代码位于 C 层级，然后更新到相应地手机屏幕。

用户接口包依赖于 OpenGL 三维库，创建一个简单的 3D 世界要有以下三个对象：均匀的光源，绘制 VGA 帧缓冲的墙，墙前的一个视角。UI 包第一周期把 VGA 帧缓冲中的内容绘制到强上，然后，在这个视角的用户，坐在墙的前面，从墙上看到的虚拟机的 VGA 输出，只是类似于影院观众观看电影。3D 对象的“纹理”属性，会通过绑定虚拟机的 VGA 帧缓冲的指针定期刷新。

为了更好的性能，会创建一个专用线程，不断更新 VGA 帧缓冲器上壁，同时运行 OpenGL 渲染脚本生成可视场景。帧缓冲获取的步骤，更新和渲染在图中示出 5。

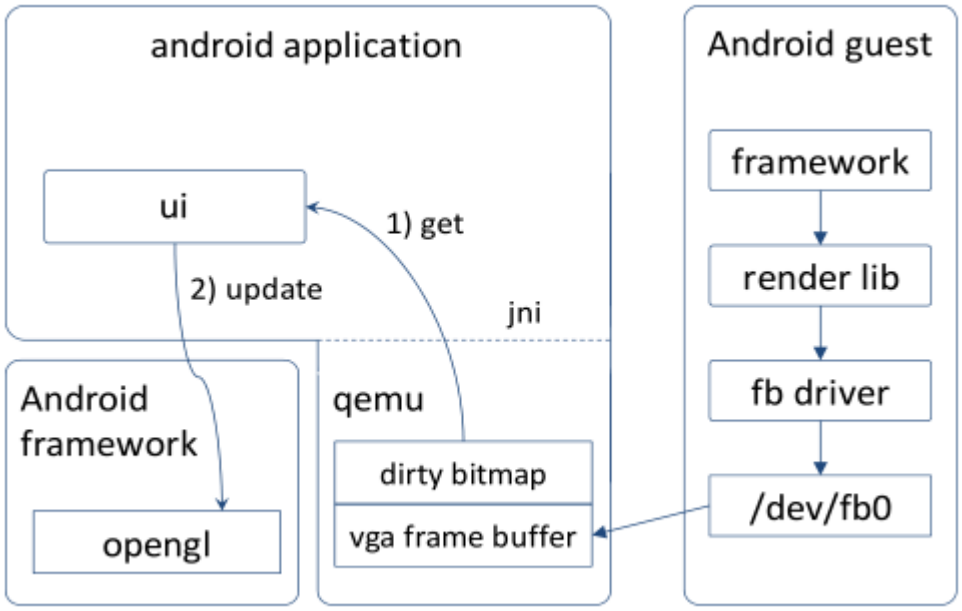

图 5 显示加速度的设计

由于大部分时间的虚拟机的显示是不变的，更新和渲染不变帧的活动应该跳过获得更好的性能和效率。作为 QEMU 支持脏记忆追踪特征，我们进一步利用这个特征提高展示体验。如果 VGA 帧缓冲器未标记为脏，其更新被跳过。否则，当前帧被更新到 OpenGL 中，然后由复位存储器脏标志重置渲染后续操作。

由于脏内存跟踪的 API 的实现基于硬件 MMU 的写作保护功能，追查内存脏成本相对便宜而跳过帧更新的结果是可以显著降低 CPU 和 GPU 的负载。

试验结果

在本文中，虚拟化智能手机的工作原型已成功建成。利用 Android-x86 项目[13]，一个开源项目该端口的 AndroidAOSP 图像基于 x86 架构的机器，并通过台湾软件工程师维护，作为虚拟机的客户 OS，运行

在商务手机 ASUSZenfone2 上。该手机已 root 和解锁引导程序，我们的项目具有关联方密切合作，并积极支持。总之，我们的项目利用 QEMU，一个开源项目管理程序，并将 SVMP 用于传感器和 I/O 重定向机制。最后，我们显示加速机制显著改善显示开销的时间。

当前原型支持的功能包括：

（1）OpenGL 直接显示。通过直接访问并显示客户 VM 的帧缓冲。

（2）单点和多点触控。多点触摸事件能够被转发到 VM。

（3）屏幕旋转事件。当主机屏幕旋转时，该 VM 将也回转它的屏幕。

（4）拨出电话。在尝试从 VM 内拨打一个号码，用户将被重定向到主机手机中的拨号应用里。

（5）GPS 位置。在主机 GPS 数据的每一个变化将被发送到虚拟机，使虚拟机和主机手机呈现类似的行为。

（6）音频重定向。该 VM 能够直接使用主机的音频 HAL（硬件抽象层）播放声音。

我们显示加速度能够达到 27～28FPS（帧每秒）的平均水平。再加上上面提到的六个特征，这主要是使我们脱颖而出的因素以及在给我们带来相同的位置于现有的间接竞争对手（如三星，VMware 的 MVP）。此外，正在进行的工作正在做，以提高对三维图形支持和加速 VM

Table 1 Host Overhead Measurement

Overhead of the host when	Memory (MB)	Energy (mAh) per 10 min	Current (mA)
(a) KVM turned off	854.81	3.046	18.28
(b) KVM turned on with no VM is running	858.37	3.146	18.88
(c) KVM turned on with one VM is running	866.12	17.06	102.36

表 1：主机开销测量

表 1 给出宿主机在 KVM 开启前后和 VM 是否运行情况下的内存和电量的开销。

内存使用的范围是围绕 1181 至 1308 MB 而电流指示从情况（a）至（c）的增量。在情况（c），

VM 是运行中，展示了当前绘制消耗 102.36mA，但上述（a），这是基线早已用完了 18.28 毫安。因此，主机绘制增量在当前是 102.36-18.28=84.08mAas，这是运行一个虚拟机的影响。

Table 2 Individual Overhead Measurement

Individual Overhead	Memory (MB)	Energy (mAh) per 10 min	Current (mA)
(a) A browser app	270	54.4	326.4
(b) An idle VM	566.5	31.25	187.5
(c) A VM running an active browser app	568.5	72.55	435.3

表 2，单独的开销测量

在主机上运行的浏览器，空闲 VM 和运行中浏览器的 VM 的比较如表 2 所示。我们的原型中运行一个 VM，读者有一些直观的开销估计。如在第二列中，VM 的内存使用可以根据初始分配而变化，在在这种情况下，它最初分配 512MB。有趣的是，情况（a）和（c）给了我们运行浏览器应用程序，和运行 VM 中的浏览器的比较，比较可知运行的 VM 仅需要比在运行浏览器需要多大约 33％的资源。

Table 3 Virtualization Storage Footprint

	KVM module (kB)	QEMU executable (kB)	Zenfone2 kernel w/o KVM (kB)	Zenfone2 kernel w/ KVM(kB)
Disk Usage	896.1	32,710	12,816	13,047

表 3，虚拟化存储空间

如表 3 所示，有我们的虚拟化技术产生的四大存储空间。第一列是 KVM 模块占用的总空间，即，包括 kvm_intel.ko 和 kvm.ko.下一列，

修改的 QEMU 可执行文件，以运行 Android 中环境中的每个 VM 映像，其占用 32.7MB 磁盘。该尺寸相对较大，由于移植工作中将 QEMU 从 Linux 移植到 Android 环境，使一些动态库被制作成一个静态的可执行文件。重要的是，最后两个栏显示 Zenfone2 内核映像的大小，分辨在 KVM 构建到内核之前和之后。

虚拟移动基础设施

设计原则与架构

VMI 的主要原则是移动虚拟机驻留在远程服务器和用户连接到虚拟机然后通过网络回传屏幕。因此，在 BYOD 解决方案里，客户端的智能手机不会遭受到运行 VM 或运行某些设备管理进程的巨大开销。（注：这里比对的是 smart-rich functionsmartphone 的方案）

利用 SVMP（安全虚拟移动平台），一个免费的开源项目，我们的

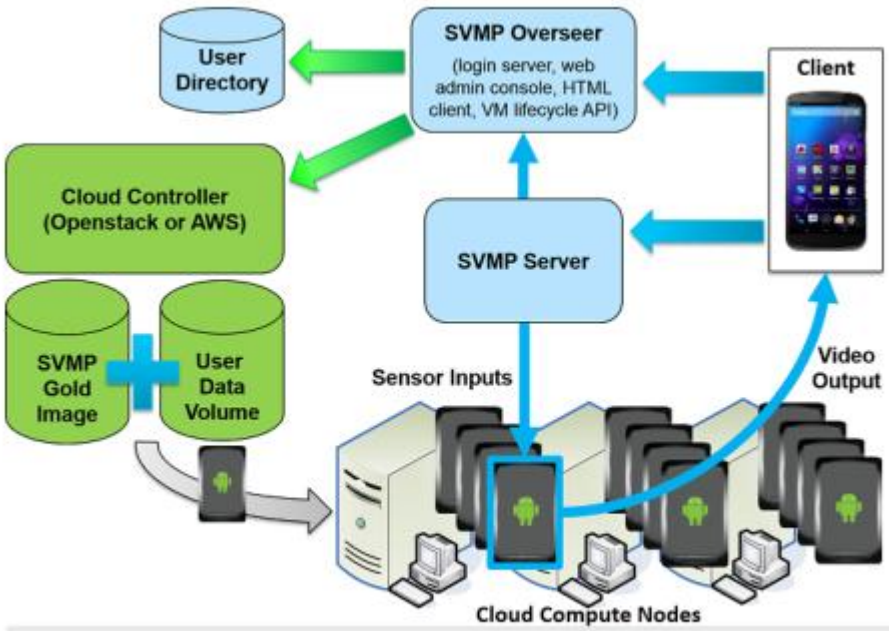

Figure 2 SVMP VMI Architecture

早期工作 VMI 的原型已经实现。图 6 示出的结构和工作流。首先，客

户端通过本地手机 app，连接到在 SVMP 巡察，通过认证和随后的连接被重定向到作为代理的 SVMP 服务器，再次重定向连接到指定的虚拟机。最后，连接是被运行在 VM 内的 SVMP 守护进程接收。一旦到虚拟机建立连接时，VM 通过的 WebRTC（网络实时通信）协议发送的视频流。

此外，SVMP 还可以在现有虚拟化系统和公共/私有云（如 OpenStack）之上轻松部署为应用程序

实现细节

与为键盘和鼠标输入设计的传统远程桌面应用程序不同，SVMP 允许用户使用多点触控，位置和传感器等本机移动输入自然地与远程应用程序进行交互，以便客户端应用程序和远程 VM 之间更好地交互。因此 SVMP 添加了一组虚拟输入设备，视频流输出，以及其他一些自定义设置实现丰富的远程访问体验。更多细节显示在图 7。

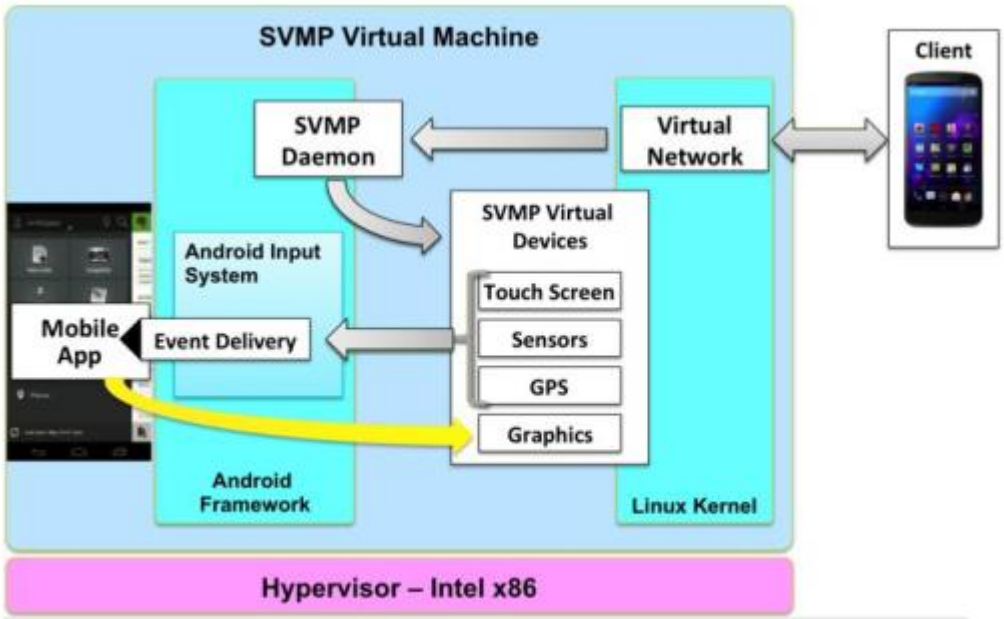

Figure 3 SVMP Virtual Device Structure

SVMP 虚拟设备架构

如图 7 所示，修改 VM（Androidx86）的源代码的是必须的，可以实现远程客户机和虚拟机之间的更自然，丰富和用户友好的交互。该 SVMP 守护进程，触摸输入，传感器，位置更新，意图和通知，视频是需要修改的组件。

（1）该 SVMP 守护进程是在虚拟机中后台运行的服务，是客户端用户输入到 VM 的主要进入点。

（2)客户端手机所生成的触摸输入事件被使用协议缓冲器转发到 VM，通过将它们注入到 VM，由 SVMP 守护程序处理接收。

（3)在客户端应用程序生成的传感器事件，是使用协议缓冲区由 SVMP 守护程序转发到虚拟机上的本地监听套接字。然后，SVMP HAL 模块 libsensors 监听 socket 和处理实际传感器事件。

（4）在客户机和虚拟机中的地点事件传达，使用的都是是协议缓冲

消息。从虚拟机上的应用向 LocationManager 请求的任何订阅意图，会被传递回客户端应用程序。然后客户端回传位置信息到 VM。

（5)在 SVMP 客户端应用程序和虚拟机之间的意图的交流是通过 SVMP 支持。例如，当用户试图调用 VM 中的打电话，SVMP 客户端应用程序能够接收 ACTION_DIAL 意图。此外，从 VM 收到的通知也将被从客户端显示应用程序。

（6）VM 的视频输出，在最低层级，将显示给来自 Linux 内核的虚拟帧缓冲设备（VFB)，而不是一个真正的视频设备。屏幕已完全更新后复制被写入到 VFB 的帧，然后为视频逐帧的编码和流媒体喂入(fed to)WebRTC 的子系统。在接收端，来自 VM 的流媒体源简单地显示作为标准的视频数据流源。因此，客户端应用程序把它作为一个标准的 WebRTC 视频流来处理即可。

试验结果

我们的SVMP项目实验,通过使用已经整合了SVMP守护程序的Android x86VM 镜像完成的。其结果是相当不错的。在功能方面，我们已经验证了多点触摸，屏幕旋转，GPS 传感器转发和拨打电话。在的流畅度方面,它仍然高度依赖上互联网连接速度。此外,一些修改,如假 IMEI 和 WiFi 的 MAC 地址数提供商已实现。然而，仍然有很长的路要走，比如 3D 虚拟 GPU 支持或直通仍然是一个具有挑战性的问题，以及如何使虚拟机看起来像物理的手机仍是我等之挑战。

结论

在本文中，吾等已实现可用于商业化之虚拟智能手机原型，并测量性能，有趣的是，接近真实的用户体验，使得它被认为是，提供商业级虚拟化 BYOD 智能手机是足够成熟的。从 SVMP 项目派生的虚拟移动基础设施的原型，描绘了一个几乎完备的框架，可以在今日交付瘦客户端的 BYOD 解方案。使用智能手机虚拟化技术，我们正在努力使品牌的移动设备供应商的方式来切入企业 BYOD 安全市场，在今天可用的单一 UI 和定制 app 之外，创造独特的智能手机功能。未来的工作，我们将专注于在 VM 或者非虚拟化的 arm 设备中最大化可运行的 app 数量，以确保用户获得和原生手机完全一样的体验。举例来说，GPU 虚拟化仍不适用于 AndroidVM;实时多媒体流，如相机，语音传感器或直播在大多数网络基础设施，有限的终端设备的计算和存储器资源存在挑战性问题（2）。

参考

[1]EldenNelson，“调查显示，BYOD 和云是最重要的数据泄露事件和恶意软件的风险所在“ https://www.csoonline.com/article/2906359/byod-and-cloud-are-top-data-breaches-and-malware-risks-survey-shows.html

[2]Ji-EunLee，Se-HoPark 和 HyoseokYoon，“支持各种移动操作系统的基于安全策略的设备管理”，计算技术与信息管理（ICCTIM），2015 年第二届国际会议日期，第 156-161 页，2015 年 4 月 21 日至 23 日。

[3】V。Munshi，Virtualization：Concepts and Applications，ICFAI University Press，2006。

[4]“Virtualization”[online]。Available：
https://en.wikipedia.org/wiki/Virtualization
https://zh.wikipedia.org/wiki/%E8%99%9B%E6%93%AC%E5%8C%96

[5]陈晓怡，“Smartphone virtualization：Status and challenges，“Electronics, International Conference on Communications and Control（ICECC），第 2834。-2839。

[6]ShakuntalaP.Kulkarni1，SachinBojewar 教授，“Survey on Smartphone Virtualization Techniques”国际研究杂志工程调查和技术（IRJET），第一卷。02，Issue：04，pp.371-376，July-2015。

[7]贾斯汀马斯顿，”虚拟移动基础设施：安全的数据和应用程序，在代替的该设备，”https://www.networkworld.com/article/2937789/virtual-mobile-infrastructure-secure-the-data-and-apps-in-lieu-of-the-device.html Jun18，2015 年。

[8]EricY.Chen 和 Mistutakaltoh，“虚拟智能手机在 IP，”IEEE 国际研讨会上无线移动多媒体网络的世界（WoWMoM），2010 年，第 1 页-6。

[9]MasashiToyama ， ShunsukeKurumatani ， JoonHeo ， KenjiTerada 和 EricY.Chen ，“Androidasaserverplatform，”IEEEConsumerCommunicationsandNetworkingConference（CCNC），2011，pp.1181-1185。

[10]“虚拟在云智能手机“[线上]。可用：https://svmp.github.io/index.html

[11]“SVMP 系统设计与实施架构”

[线上]。可用：https://svmp.github.io/architecture.html

[12]“开源 SVMP 项目”：
https://github.com/SVMP

[13] Android-x86.org
http://www.android-x86.org/

作者：卓傅育，林浩澄，洪茂荣。发表于 《电脑与通讯》第 165 期。