More Mobile than Mobile

Will Dietz, Kevin Larson, Shivaram Venkataraman
{wdietz2, klarson5, venkata4}@illinois.edu
University of Illinois at Urbana Champaign

1 Introduction

Mobile devices today run many third party applications to perform complex tasks like web browsing, banking and gaming. Recent studies have found that smart-phones are the target of an increasing number of malware attacks [12], [3], [4] and their security is important as personal data such as contacts, credit card numbers and passwords are often stored on the device. While some security models [2] provide a stronger process level isolation among applications, operating system bugs such as [8], [6], [7] allow malicious applications to take over the device. We believe that virtualization can be useful for secure isolation of third party code from confidential data and provide a greater defense-in-depth against attacks on the system.

In recent years, virtual machines have become prevalent in cluster computing environments [1] as they lower power costs and help in conserving data center space. Hardware improvements have meant that smart phone configurations found today resemble desktop machines from few years ago and many of them run commodity operating systems. There is a growing interest in academia [14] and industry [5] about the benefits of virtualization on these devices. We believe that virtualization can provide better security guarantees in mobile devices and enable useful applications like environment migration.

Environment migration has been studied earlier, in the context of servers in a cluster [13] and enables administrators of clusters to perform maintenance tasks without interruption. On the other hand, migrating a system to a mobile device can take advantage of network or computation facilities that are closer to the user's location and provide the user with a consistent experience irrespective of the network connectivity. Migration techniques also help maintain consistent snapshots which allow easy transfer of data when users switch mobile phones and to roll-back the system to a previously known state.

NOTE: Need to mention OK-L4 and ARM Trustzone

1.1 Existing Work

Currently, there are many solutions available for virtualization on desktop environments. VMware is a popular closed source solution which implements a variety of virtualization techniques and is used in both industry and academia. KVM [15], QEMU [11], and XEN [10] are all open source solutions, implementing their own assortments of virtualization techniques. These solutions for the most part do not work well in a mobile environment for performance, security, and usability reasons.

Recently there has been a surge of research done for mobile virtualization. One such solution is MobiVMM [17], which prioritizes performance and security at the cost of usability and portability. Work has been done to port KVM to Android [9], focusing on performance and functionality. All of these solutions either dual-boot the OS or require disabling the phone's existing runtime stack to use them.

VMware's MVP project [16] is most similar to ours. They introduce a very thin hypervisor with an emphasis on usability, performance, and security–without sacrificing the phone's functionality. However their implementation does not integrate with the host OS, but rather replaces and contains it. This is useful, but tangential to our work. As described in Section 1.2, we aim to provide a Type II VM and prioritize live migration capabilities as well as usability. Furthermore, we are mostly concerned about containing applications for isolation, security, usability, and portability. Instead of virtualization the existing OS to protect others from it, we assume it is trusted and only have to isolate new applications. This provides for a very different architecture in our implementation.

NOTE: Add some discussion about power usage, memory constraints ?
NOTE: Should we mention OP (like Prof King mentioned, citing papers of people reviewing it never hurt ;)) but also because are architecture, accidentally, does share a lot with the OP design...?

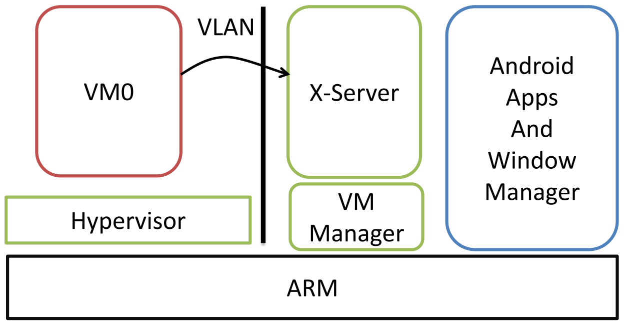

Figure 1: Architecture diagram

1.2 Proposed Architecture

Our proposed architecture has two main components: the virtualization framework, and the integration front-end. The virtualzation framework contains the hypervisor (QEMU or KVM based) as well as each of the spawned VM's (see Figure 1. Each VM will contain a thin OS (as thin as possible) with the primary goal of each VM running an application. The VM's will be running x86-based Operating Systems as a design decision to enable a more diverse set of target applications to be run (note that most mobile devices that we are targeting are ARMv6 or ARMv7). Each of the VM's will headless themselves, but will run their applications via X, connecting to the X server.

The other component is the integration front-end. This contains a light X server that has been integrated into the host OS, and also contains the VM manager. The VM manager will either run inside the X-server as a native application (ARM-based), or as a separate application within the host OS. The VM manager will be the user interface that controls launching, suspending, resuming, migrating the VM's as well as ensuring the X-server is running properly. Note that most of this actual logic and functionality will be in the hypervisor in the virtualization framework.

Note that this shared X-server allows us to make the graphical part of the applications common, which we believe is an improvement over having each of the VM's run an X-server, and then have another layer compositing or managing those into what the user sees. Additionally the X-server will be running native code, which we hope will allow for it to be less cpu-intensive than it might otherwise.

Additionally we aim to have the slowest part (the virtualization framework) be as device independent as possible, which is important because we anticipate this being the most difficult to make effective. The integration front-end will have to be ported for each new device we support, but we hope to make that as simple as possible. As an example, we should be able to take advantage of existing desktop X servers and support desktops as a client for our system, but we leave that as future work. Our first implementation will focus on one particular device and OS, but even there we will be able to support live migration of applications.

2 Timeline

2.1 Feb 17 - March 17

- Explore architecture details of ARM and finalize design details for the hypervisor.
- Build a thin x86 hypervisor on ARM using the mobile device toolchain.
- Compile and run x11 on the mobile device.

2.2 March 17 - April 27

- Integrate x11 with Window Manager of the mobile device.
- Implement VM Manager to spawn the virtual machines and x11.
- Implement migration UI on desktop and mobile device.
- Migrate existing x86 applications between desktop and mobile device.
- Investigate performance improvements to make emulation faster.

2.3 April 27 - May 11

- Document appropriately and write up the final report.

3 Evaluation plan

We evaluate our system focusing on our design goals of portability, isolation and usability

- Running existing unmodified x86 binaries mobile device.
- Measure performance impact of x86 emulation on ARM.
- Evaluate the latency of an interactive application which renders using the native X-server.
- Profile the memory and power overhead due to virtualization.

4 Anticipated results

- Working implementation of x86 Hypervisor on ARM

- VM Manager integrated with the mobile device window manager

References

[1] 16 percent of workloads are running Virtual Machines. `http://www.gartner.com/it/page.jsp?id=1211813`.

[2] Android Security and Permissions. `http://developer.android.com/guide/topics/security/security.html`.

[3] Cyber-criminals target mobile banking. `http://www.v3.co.uk/vnunet/news/2173161/cyber-criminals-target-mobile`.

[4] iPhone Privacy. `http://seriot.ch/resources/talks_papers/iPhonePrivacy.pdf`.

[5] Mobile Phones, The Next Frontier. `http://blogs.vmware.com/console/2009/08/mobile-phones-the-next-frontier.html`.

[6] Vulnerability Summary for CVE-2009-0475. `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-0475`.

[7] Vulnerability Summary for CVE-2009-2204. `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-2204`.

[8] Vulnerability Summary for CVE-2009-2692. `http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2009-2692`.

[9] D. A. Andreas Nilsson, Christoffer Dall. Android Virtualization, 2009.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization, 2003.

[11] F. Bellard. QEMU, a fast and portable dynamic translator. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 41–41, Berkeley, CA, USA, 2005. USENIX Association.

[12] A. Bose and K. Shin. On Mobile Viruses Exploiting Messaging and Bluetooth Services. *Securecomm and Workshops, 2006*, pages 1–10, 2006.

[13] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, page 286. USENIX Association, 2005.

[14] L. Cox and P. Chen. Pocket Hypervisors: Opportunities and Challenges. In *Eighth IEEE Workshop on Mobile Computing Systems and Applications, 2007. HotMobile 2007*, pages 46–50, 2007.

[15] Qumranet. Kernel-Based Virtual Machine. [Online], 2009. Available: `http://linux-kvm.org`.

[16] VMware. VMware MVP (Mobile Virtualization Platform). `http://www.vmware.com/products/mobile`.

[17] S. Yoo, Y. Liu, C.-H. Hong, C. Yoo, and Y. Zhang. MobiVMM: a virtual machine monitor for mobile phones. In *MobiVirt '08: Proceedings of the First Workshop on Virtualization in Mobile Computing*, pages 1–5, New York, NY, USA, 2008. ACM.