Multipersona Hypovisors: Securing Mobile Devices through High-Performance Light-Weight Subsystem Isolation

Neelima Krishnan
Computer Science, Virginia Tech
Blacksburg, USA

Seth Hitefield
ECE, Virginia Tech
Blacksburg, USA

T. Charles Clancy
ECE, Virginia Tech
Blacksburg, USA

Robert W. McGwier
ECE, Virginia Tech
Blacksburg, USA

Joseph G. Tront
Computer Science and ECE, Virginia Tech
Blacksburg, USA

***Abstract*— We propose and detail a system called multipersona Hypovisors for providing light-weight isolation for enhancing security on Multipersona mobile devices, particularly with respect to the current memory constraints of these devices.**

Multipersona Hypovisors leverage Linux kernel cGroups and namespaces to establish independent process container, allowing isolation of the Multipersona process tree from other simultaneous instances of Multipersona and the hypovisor which is an underlying Angstrom-based embedded Linux distributions designed to add additional security to the system. The system incorporates a wide range of data integrity tools in the embedded hypovisor, and an SE Linux-enabled kernel for mandatory access control and integrity tools for transparent auditing of running Multipersona instances.

A prototype is presented which uses integrity tools external to the Multipersona container to audit it for malicious activity, and also has the ability to support a multipersona environment with multiple encrypted personas existing individually or simultaneously on the device. Two versions are demonstrated, one which allows cold-swapping of personas for high-assurance scenarios and also one that supports hot-swapping.

Analysis shows that the hypovisor has a 40-50 MB impact on the overall memory footprint for the system.

***Keywords*—** Mobile Device, Light-Weight Virtualization, SE Linux, Security, Access Control, System Policy, Multipersona

I. INTRODUCTION

Mobile optimized applications and cloud services have transformed the computing market from desktop based legacy applications to an entirely new environment of apps that access data stored in the cloud. There has been an explosion of low-cost mobile devices that have revolutionized the computing ecosystem. Mobile devices have become indispensable, convenient tools because they offer increasingly large capacity in fast, easy to use, compact, portable form factors.

Application markets provide central points for application distribution and discovery. Markets such as Apple's App Store and Google's Play remove barriers of entry for developers by simplifying sales and distribution. Combined these platforms with relatively easy to use application programming interfaces, the markets are lush with millions of applications. On the consumer front, markets simplify discovery, purchase, and installation of applications. This process is self-contained on the handset.

While these mobile devices offer tremendous opportunity to enhance connectedness and productivity, they also introduce a broad new range of security challenges to enterprise networks, personal networks, and broadly to the Internet. Mobile devices far exceed desktop PCs as endpoints, and offer much less sophisticated infrastructure for providing device security. Many enterprises rely heavily on secure end-points to bootstrap the security of their entire network, and these mobile devices lack the required security controls to operate securely in a variety sensitive domains.

A wide variety of research, development, and commercial products seek to address fundamental gaps in mobile security. In this paper we focus specifically on the security of the underlying mobile operating system and seek to improve its fundamental integrity. We focus on the Multipersona mobile operating system (OS) because it has the largest market share among mobile multipersona operating systems, and its open source nature allows for easier experimentation than a variety of its competitors. In this paper, we develop a variety of kernel and OS layer security features for Multipersona.

Our contribution focuses on improving core operating system (OS) security for Multipersona through a number of novel components:

1. Linux cGroups are used to run Multipersona inside a container (i.e. isolated process namespaces) that provides isolation between Multipersona and an underlying embedded Linux distribution that uses a variety of open source integrity tools to audit the running Multipersona instances, known as the hypovisor.

2. A multipersona application is described where an operating Multipersona container is shut down and a secondary secure container boots up from an encrypted file system stored on a removable SD card. While this cold-swap scenario offers some usability issues, it sup-ports a high assurance environment where simultaneous execution of code at different security levels is generally not permissible.

3. An architecture based on the Cells project [2] is presented

for multipersona scenarios where simultaneous instances of Multipersona in parallel containers are able to operate on a single device using a variety of virtual hardware multiplexers.

4. SE Linux policies are developed to provide integrity to the Linux kernel and hypovisor, further extending the SE Multipersona project [21].

The reminder of this paper is organized as follows. Section 2 provides background information on Multipersona, its security challenges, and prior work in the field. Section 3 details the Multipersona Hypovisors architecture. Section 4 details the system implementation and gives an analysis the memory footprint impact. Sections 5 outline ideas for future work and conclude.

II. BACKGROUND

In this section we provide an overview of the Multipersona security ecosystem, including an introduction to the Multipersona security model and an outline of different security threats to that model. We then provide a summary of major research in OS-layer security for Multipersona to put our contributions into context.

A. Security in the Multipersona Ecosystem

Multipersona is best described as a middleware running on top a Linux kernel and set of non-GNU shared libraries. It pro-vides a common infrastructure and Java-based virtual ma-chine for apps to execute. When compared to the other leading multipersonas, Multipersona multipersonas are generally less expensive. There is an active community of developers and multipersona enthusiasts who develop and distribute their own version of the OS. There are millions of applications freely available from the Internet, which makes the market place quite broad.

This convenience bears with it some associated risks. Mobile devices can easily be stolen or misplaced. Either misfortunes result in breaches of confidential information whether or not the information contained in them is accessed. Beyond this, privacy breaches can also occur as a result of utilizing unsecured wireless networks, installing apps that harvest information, or users sharing credentials with untrusted services.

Significant research is being conducted in the area of mobile security, to ensure the privacy of both a user and their data. Research progress in personal information management on mobile devices, specifically multipersonas, is growing at exponential rate in the area of human computer interaction. This enables user to have their confidential information available in various personal devices simultaneously [11].

Multipersona security solutions have been proposed and implemented up and down the hardware/software stack, from hardware roots of trust to application layer antivirus and Data-loss prevention tools. The growing consensus is that any integrated device will leverage layers of security for a defense in depth strategy.

In general, we seek to achieve security goals of confidentiality and integrity for data stored on a mobile device (note that confidentiality is distinct from privacy, as many users intentionally choose to share potentially sensitive information through social networking apps). We seek to mitigate threats to confidentiality and integrity, to include more advanced usurpation threats. The threat model assumes an adversary is able to execute arbitrary code on the device as a standard user, and may potentially leverage privilege escalation attacks to gain administrative access. The most likely vector for running hostile code on the mobile device is through malicious apps masquerading as legitimate software on an app market.

Our contribution focuses at the OS level, and includes aspects of the kernel, OS, and Multipersona middleware. These tools generally need to be baked in "when the device firmware is developed, and therefore are not generally viable as after-market security tools installable by a typical user.

B. Multipersona Security Model

Security in the Multipersona OS is implemented using the concept of a secure sandbox [5]. In the context of computer security, sandboxing refers to a method of separating running programs. That is, no application by default has permission to perform any operation that would impact another application, the OS, or the user. This includes actions such as writing or reading private data (e.g. contacts, e-mails, and home screen), network access, affecting the device sleep state, or accessing another application's files. Separating running programs creates a confined execution environment, which helps isolate problems and with individual applications.

Multipersona is a fully multitasking OS and uses the inherent Linux model of groups, users, and signature verification for executable files. The Multipersona framework accomplishes sand-boxing by assigning each application a distinct Linux user ID (UID) and group ID (GID). The Linux kernel, which is the foundation of the Multipersona system, uses the separate UID/GID to provide isolation between applications.

Since each application and its corresponding data have unique UID/GIDs, other applications cannot gain access to them unless explicitly stated. Any application that wants access to another's data or global resources, such as net-work access, must request the corresponding permissions from the system at install time through the MultipersonaMan-ifest.xml file. It is the user's responsibility to evaluate the application's requested permissions and approve or deny its installation onto the device.

The developer uses a unique certificate to digitally sign the installation file package. During installation, the system displays the application's requested permissions to users who can either proceed or cancel the install. If these permissions change at any point after installation, the app's digital signature will no longer match, and the application will be blocked. If an application attempts to access

resources without the corresponding permission, whether by a bug or a user with bad intentions, it will be force-closed and the security breach which recorded in the system log.

C. Security Issues

A problem with Multipersona's security framework is the lack of built-in, fine grained control over an application's access capabilities. An excellent example of this issue with the An-droid permission scheme is a known malicious application, iCalendar [19]. The iCalender app is used to record daily events and also synchronize them with the user's inbox in order to send reminders. Thus, iCalendar will need access to both the Internet and the coarse location. The app requests the corresponding INTERNET and ACCESS COARSE LOCATION permissions, and in addition it requests RECEIVE SMS and SEND SMS. Closer examination of its source code shows that iCalendar covertly sends a text message to a number on the fifth click on the phone. It also blocks all incoming messages originating from the destination number. By intercepting incoming messages, iCalendar can conceal itself and the user will never know that anomalies had occurred on their phone.

Anyone can upload an application in the Multipersona market by simply paying a fee of $25 USD, with his own digital certificate. There is no code inspection [13]. The application developer digitally signs his app. There is no Certificate Authority who verifies the authenticity of the app's signature. Applications can be downloaded and installed from non-market place like piratebay.org, eBay, fileshare.org, etc.

Many malicious applications find their way into smart-phones through these vectors. A more insidious class of applications are able to execute privilege escalation attacks [6, 9]. By obtaining administrative privileges on the underlying OS, apps can gain access to the environment of any other apps, by breaching the sandbox environment.

Another attack scenario is maliciously colluding applications [17]. Users rarely evaluate an application name and decide if it appears legitimate. Some users assume the downloaded application is a well know application, without reading the permission list. Examples are Facebook, Google+, Yahoo Mail, and Gmail which are downloaded from non-trusted marketplaces.

D. Previous Work

Here we investigate previous approaches to providing security to the underlying Multipersona OS and its services. They range from kernel-level tools to those that integrate within the Multipersona middleware itself to provide a variety of security services.

At the kernel level, integration of SE Linux into the An-droid offers the ability to significantly improve OS integrity. Early work [18] demonstrated that the Linux kernel sup-porting Multipersona could be rebuilt with SE Linux enabled, and that user space tools could be cross-compiled using existing tool chains. This work was further extended to develop a broad range of security policies for Multipersona and released open-source by the National Security Agency as the SE Multipersona project [21]. The SE Linux work performed under this project has been ongoing for nearly two years, was contemporaneous to the development and release of the SE Multipersona project, and has since been merged with the open-source distribution for consistency.

Use of independent namespaces for process isolation was first introduced by Cells [2]. Cells is a virtualization architecture for enabling multiple, virtual multipersonas to run simultaneously on the same physical device in an isolated manner, but did not seek to formally address device security. Cells introduce a usage model of having one foreground virtual phone and multiple background virtual phones. This model uses a device namespace mechanism and device proxies that integrate with OS virtualization to multiplex hardware while providing native hardware device performance. Virtual phone features include fully accelerated 3D graphics, complete power management features, and full telephony functionality with separately assignable telephone numbers and caller ID support. A prototype implementation supports multiple Multipersona virtual phones on the same phone. Performance test results demonstrate that Cells imposes only modest runtime and memory overhead, works seamlessly across multiple hardware devices including Google Nexus 1 and Nexus S phones, and transparently runs Multipersona applications at native speed without any modifications.

A wide variety of research projects and deployed products seek to provide threat detection and mitigation at either the middleware-layer or application layer on Multipersona devices. One example is TaintDroid [8] which marks memory and storage locations as sensitive information propagates through the system to identify privacy leaks. Another example is XMmultipersona [4] which monitors inter-process communication to identify potential privilege escalation attacks.

Our approach leverages the prior work from the SE An-droid and Cells projects as building blocks for creating an OS with true defense in depth. We focus on the development of the hypovisor which provides the extended integrity tools, and layer in the isolation, SE Linux, and multipersona tools for an integrated solution.

III. MULTIPERSONA HYPOVISORS

Figure 1 shows the Multipersona architecture. Multipersona utilizes the open-source Linux kernel that permits customization for virtualization, security policy enforcement, and other hardening techniques. Our solution seeks to improve mobile device security focused on the threat to information confidentiality and integrity. The solution supports a variety of use cases:

1. A single persona exists on the device, is instantiated

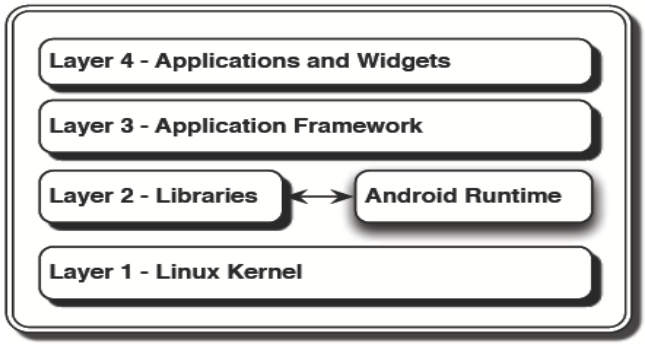

Figure 1: Basic Multipersona Architecture

within an isolated container, and secured using the hypovisor and SE Linux;

2. Multiple encrypted personas exist but only one is active at a time, for high-assurance environments, and are secured using the hypovisor and SE Linux;
3. Multiple simultaneous personas operate on the device in independent containers, and are secured using the hypovisor and SE Linux.

The integrity of the simultaneous multipersona system is based entirely on the ability to break out of one namespace and swim upstream to a parent namespace. To accomplish this, the ability for an adversary to obtain root permissions and exploit the underlying kernel in any of the container must be mitigated. SE Linux helps greatly by minimizing the kernel accesses available to a root user in a specific namespace. However, zero-day vulnerability may exist that allows an adversary to exploit the container. Consequently for high assurance environment where we assume an adversary is able to obtain root on a non-secure container, the cold-swap approach is preferred, as its integrity of the secure profile is cryptographically protected.

A. Containers Infrastructure

This section further details the components of the overall containers infrastructure necessary to implement the overall system architecture, which is depicted in Figure 2.

1) Linux Containers

The cGroup and namespaces features of the Linux kernel are used as mechanisms for high-performance, multipersona container virtualization. This allows multiple, simultaneous, isolated instances of Multipersona on a single device to support multiple independent security domains. When combined with chroot, they allow booting multiple simultaneous init processes from independent root filesystems that are isolated from each other, all operating on top of the same Linux kernel. However, given the dev entries will likely share the same major/minor numbers, there may be contention for hardware resources if multiple, simultaneous instances are booted without resolving hardware drivers.

This mitigates a large portion of potential attack vectors against the system, making it nearly as effective as bare-metal virtualization approaches. Current approaches to devices supporting multiple simultaneous security domains

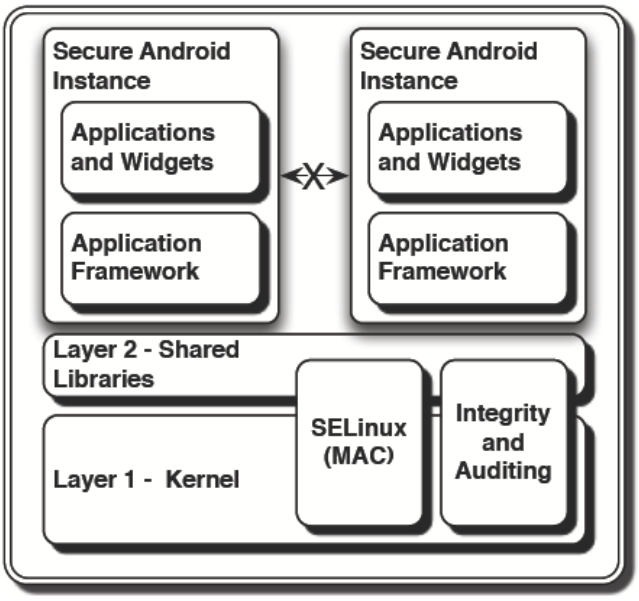

Figure 2: Proposed Multipersona Architecture

require full bare-metal virtualization, meaning that for N simultaneous security domains, the device must execute N simultaneous Operating Systems and N simultaneous middleware instances. This seriously affects device performance in resource constrained mobile devices designed to support only a single OS instance. By using containers, common device functions can be shared by a meta-domain, requiring only a single shared instance to be executing at a time.

As long as root privileges can be sufficiently restricted, it should not be possible to break out of a container and access data and processes from another container. However if the integrity of the kernel is compromised, an attacker can cross security domains. To combat this, we pair containers with SE Linux and author SE Linux security policies that protect the light-weight virtualization subsystem and kernel from attack.

2) Light-Weight Hypervisor

A current constraint of mobile devices is that a user can disable any security software with administrative access to the device. A hypervisor is a subsystem that allows for external auditing of system integrity, executing below the main OS. In typical desktop and server environments, this is implemented by first booting a barebones Linux instance (which becomes the hypervisor), and then booting fully featured Linux instances on top.

By using containers, we are able to significantly reduce the overhead footprint by not requiring multiple instances of the kernel. We first boot an embedded Linux kernel tailored for a mobile OS. In particular, the Multipersona variant of the Linux kernel includes a variety of Multipersona specific patches, to include extended power management features. Once the embedded distribution loads, the user OS is booted using virtualization provided by containers (e.g. cGroup, namespace, and chroot). This completely isolates the main system from the embedded Linux hypervisor.

To distinguish the role of our management plane from a traditional virtualization hypervisor, we term our embedded Linux instance a hypovisor, because it sits under the various booted Multipersona instances, rather than the over notion conveyed by traditional supervisor/hypervisor terminology. Our hypovisor architecture is depicted in Figure 3, where the process tree for a variety of independent containers is shown.

Hypovisors have major advantages over traditional virtualization approaches since the hypovisor distribution is transparent to the mobile OS, and no special device drivers are required. Within the hypovisor we are able to implement a large range of device integrity tools, including kernel integrity, OS integrity, mobile middleware integrity, firewalls, intrusion detection systems, and intrusion prevention systems. In addition, administrators could use the hypervisor to easily manage virtual instances either locally or remotely, and with or without the user's knowledge or permission. In order to provide a layer of security, Multipersona is not used as the hypervisor and virtual machine host for several reasons.

First, in a typical OS virtualization implementation, the root OS can be used as one of the virtual instances. This could be a possible vulnerability for security oriented devices if the root instance were to be compromised. Compromising the host instance would, in effect, compromise all of the guest instances. To protect against this attack, the root instance is abstracted from the guests into a small embedded system that functions similarly to a traditional hypervisor. This system allows for features such as additional monitoring and enforcement of both incoming and outgoing network traffic. More importantly, the embedded hypovisor would not interact with the user (excluding switching the active virtual instance), which adds an additional layer of security to the system.

Secondly, the Multipersona system is not a typical GNU-based Linux distribution like Ubuntu or Fedora. Multipersona uses its own custom Bionic libc and Dalvik Java virtual machine. This makes porting existing embedded security tools more complicated if Multipersona is used as the host system. Also, the use of a custom embedded system reduces both the memory and storage overhead of the embedded hypovisor on the device.

3) Memory Optimization

An important aspect of working on mobile devices is recognizing the limits they present. Currently, these devices resemble embedded systems more than the traditional personal computer. The majority of devices on the market use an ARM Cortex A8 or A9 processor, 512 to 2048 MB of memory and anywhere from 1 GB to 16 GB of internal storage. In addition, they have limited supplies of power.

Because of this, it is important to remove as much overhead from the system as possible. Avoiding running multiple instances of the Linux kernel provides an opportunity to decrease the memory footprint, and is achieved by the hypovisor architecture.

However, there are many additional opportunities for memory optimization realizing that simultaneous instances of Multipersona involve significant shared infrastructure. For example, Multipersona uses the zygote process to launch apps, rather than launching each one as an independent process. The reason is because the Dalvik virtual machine loads a large footprint of shared Java libraries, and rather than every app duplicating those libraries in memory, zygote allows them to all be loaded into memory once and used among all applications.

While not demonstrated in this implementation, the ability to spawn from a shared zygote process across multiple containers would offer a novel way to further decrease the memory footprint between Multipersona instances. Implementing this would require significant extensions to Multipersona itself, ensuring sufficient rewalling between Multipersona instances leveraging a single zygote operating in a shared namespace.

B. Access Control

This section outlines access control methodologies and describes how SE Linux is used in the hypovisor architecture to achieve a higher level of assurance.

1) DAC vs. MAC

In a traditional Linux/UNIX system, access control is implemented using a discretionary access control (DAC) model. DAC is an access policy that restricts access to files, and other system objects such as processes and devices, based on the identity of users and groups to which they belong. The MAC security model differs from the DAC model in that subject's access to objects is regulated by a security policy. Mandatory access controls use sensitivity labels to determine who can access what information in the system.

2) SE Linux

SE Linux is a mechanism for enforcing fine grained MAC within the Linux kernel [21]. SE Linux is not a Linux distribution, but rather a set of kernel modifications and userspace tools that can be added to various Linux distributions. Multipersona systems built on version 2.6 and above of the Linux kernel can be protected using SE Linux. The kernel modifications make it possible to make appropriate modifications to the Multipersona baseline in order to activate SE Linux.

SE Linux enforces MAC by checking the security attributes of the object and subject against the given system's security policy. In the iCalender application example previous mentioned, the security policy of the system could be configured such that the app cannot send an SMS to an unidentified numbers over the network. Also, the SE Linux policies can be configured to prevent iCalender from gaining access to unique phone information, such as contact information or any confidential information, and sending it as the content of an SMS.

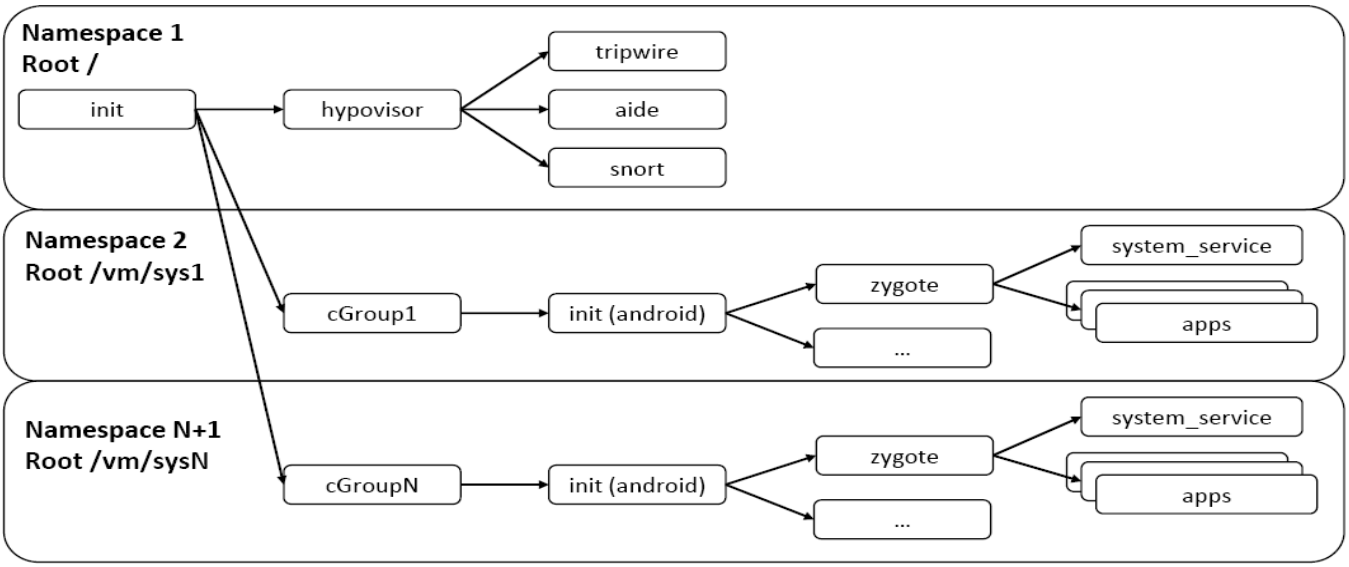

Figure 3: Process hierarchy showing Multipersona Hypovisor namespace relationships

While this example illustrates a potential application of MAC, in general it is not feasible to implement app specific SE Linux policies. MAC policies are treated as invariants across all modes of system operation, and apps involve a wide variety of context-driven access control requirements. An appropriate use of SE Linux is in the development of policies unique to the Multipersona middleware and hypovisor functions, and run all apps within a single security context.

SE Linux stores the security attributes of objects and subjects as extended attributes within the Linux kernel and file system. Extended attributes are a file system feature that enables users to associate files with metadata not interpreted by the file system. Extended attributes are natively supported by many file systems, including ext3, ext4, JFS, etc.

However, by default Multipersona utilizes the yaffs2 file sys-tem, which does not natively support extended attributes. Some open source patches have been developed in order to enable extended attribute support for yaffs2 [18]. The key features supported by the SE Multipersona distribution are per-file security labeling support for yaffs2, file system images labeled at build time, kernel permission checks controlling Binder IPC, labeling of service sockets and socket files created by init, labeling of device nodes created by ueventd, exible/configurable labeling of applications and app data directories, user space permission checks controlling use of the zygote socket commands, and use of MLS categories to isolate applications.

3) Integrity and Auditing Tools

One of the key purposes of the hypovisor is the ability to provide integrity and auditing capabilities. Through its position in the root namespace, sitting outside of but able to see into the Multipersona instances running on the device, the tools within the hypovisor can provide a broad range of security services.

It is impossible for malware to compromise a system, and attributes achieve persistence through a reboot, without altering system files. As a result, file integrity checkers are an important capability in intrusion detection. A file integrity checker computes and stores a checksum for every guarded file and during periodic verifications it recomputes checksum and compares it against the stored value to determine if the file has been modified.

For our implementation, we used the Tripwire file integrity tool [12]. Porting native Linux/UNIX applications to Multipersona can be difficult due to Multipersona's lack of the typical libc environment and other Linux utilities. In addition, installing Tripwire directly into an instance of Multipersona system would make it difficult to protect. If a malicious application were to get root access, then it could simply delete the Tripwire binaries, configuration and database.

By leveraging Hypovisors, Tripwire can be installed out-side Multipersona and within the embedded Angstrom distribution. Here it benefits both from a standard GNU build environment, in addition to isolation from malware potentially operating within subordinate Multipersona namespaces. Such malware would be unable to detect the Tripwire installation, its processes when running, or the fact it is accessing files within its filesystem.

Once Tripwire is installed, we can modify the configuration to monitor the Multipersona container's data and system directories and the init.rc boot script. Based upon this setup, we can monitor the Multipersona file system and detect the installation of different apps. Compiling the Multipersona sources with build type as “user” prevents us from having root access. This enables us to effectively test with real malware.

4) Cold-Swap Personas

The cold-swap version of the system secures a persona by encrypting the root file system of the secure personas with a

128-bit Advanced Encryption Standard (AES) key. Entering a passphrase decrypts and starts the persona. Since Multipersona separates its filesystem into several partitions, only the system, user data, and cache partitions were encrypted to optimize performance.

Three user instances were demonstrated: work use, personal use, and family use. Both the work and personal modes were encrypted. In this implementation the initial ramdisk, which contains the root mount point was not encrypted. When the user switches personas, that image's partitions are unencrypted and mounted by a loopback device onto the original system, data, and cache folders. The Multipersona system could then be restarted to load the secure persona. This allows for encrypted storage of sensitive documents, cryptographic material such as SSH or GnuPG keys, confidential information, etc.

Using multiple personas provides the ability to separate and secure data from possibly malicious applications. For example, in the secure persona, the marketplace can be disabled preventing the user from installing additional applications, which could be malware. Any applications added to the system would need to be from trusted sources. This separation of data allows users to store sensitive information, such as contact data, on the secure persona without the fear of malware in the insecure persona gaining access to that data. Also, the encryption provides another level of security if the device were lost. Because of the process and filesystem isolation provided by the hypovisor, an adversary will not know that there is a secure encrypted profile in the system.

The major disadvantage to this method is that encryption and decryption are resource intensive. In order to switch to a secure profile, the current profile must be aborted and the new persona mounted, decrypted and booted. Depending upon the strength and method of encryption used and the amount of data to decrypt, this can cause an unacceptable amount of time. In addition, keys can be lost or forgotten which would render the associated data unrecoverable.

Encryption that is managed by the user can cause problems in a managed network by rendering necessary files inaccessible to the network managers. If you forget your passphrase then there is no chance of recovering your data. In addition, only data stored on the device is protected. Remote wiping [16] of information and reset of the stolen device is possible, depending on the awareness of the user. A naive user need not be aware of the existence of such options. Also, by the time one realizes the device is stolen, and information could have been compromised.

5) Hot-Swap Personas

Use of containers allows for the concurrent execution of multiple personas with disparate security levels. This facilitates the isolation of sensitive information from different security classifications. For example, a corporation might tightly control the use of a corporate persona whereas the user's persona is more versatile to allow for other beneficial applications not adhering to corporate policy.

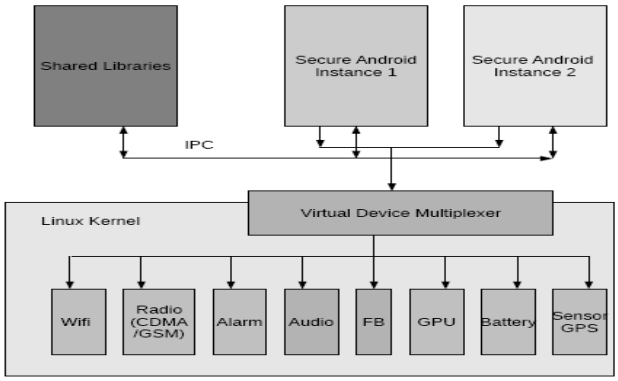

Figure 4: Device Namespace Multiplexing

With the use of multipersona virtualization, these personas can execute concurrently, avoiding the unacceptable switching costs.

IV. IMPLEMENTATION

The Cells [2] project demonstrated the power of using containers to quickly swap between different simultaneous personas. Our contribution is the addition of the hypovisor that allows these simultaneous personas to be integrity checked from below, and securing them with SE Linux. This section details the system implementation and provides a variety of quantitative performance metrics associated with Multipersona operating in a containerized environment.

A. Container Implementation

The kernel namespaces were implemented using the LXC container system for Linux. This system utilizes the built-in cGroup and namespaces of the Linux kernel to provide resource management and process isolation between the host system and the different virtualized instances [3]. The cGroup feature provides resource management (process containers) and the namespaces feature provides resource isolation for the system. In addition, cGroups is used to manage the amount of processing time each guest instance uses.

The hypovisor running the LXC container system is based on the Angstrom Embedded Linux distribution [1]. Angstrom was chosen for its foundation in the OpenEmbedded project, and broadly supports a wide variety of hardware platforms. In addition, the Angstrom distribution website provides the Narcissus online builder which allows a developer to quickly generate a custom root filesystem. The online builder provides the ability to add many packages to the root filesystem such as networking, console, and development tools. In addition, the online tool can build a toolchain used for cross compiling systems such as LXC. In order to best support the Multipersona OS within a container, Multipersona version of the Linux 3.0.8 kernel was used for the underlying system kernel. Support was added to the kernel for namespaces, cGroups, virtual networking and Ethernet.

The kernel is the core or lowest level of the system which is shared between the host and the container instances.

Table 1: Memory utilization for a variety of scenarios

	Used (kB)
Stock Multipersona	263,660
Hypovisor	37,768
Hypovisor + One Container	311,864

The hypovisor includes an embedded version of the GNU libc and busybox. Along with the generated cross compiling toolchain, these tools were used to port many existing applications to the host system.

In order to build Multipersona for the LXC container system several changes were made to the standard Google source. Specifically, the Dalvik functions used to fork and spawn new processes attempted to set the Linux capabilities for the newly created thread. This function was disabled in order work within the LXC system. In addition, some changes were made to other files such as init and init.rc.

After the changes to Multipersona were made, the Multipersona system was successfully booted in a container on the Angstrom host. Since LXC uses the namespace and cGroup features to isolate process, the Multipersona instance was unable to view any processes belonging to the host. However, the

Angstrom hypovisor host had full control over any Multipersona processes.

This allows the virtual instance to remain separate from the host or any other virtual instance. For multiple instances of Multipersona to run concurrently, a demultiplexor must be used to control each instance's access to the physical hard-ware and also to facilitate switching between instances. Figure 4 shows the how the switching can be done using a virtual device namespace multiplexer. This swaps the control of the instance and hardware devices as and when we switch between the foreground and background instances.

Summarizing, our design has the following features:

1. The kernel is the standard Multipersona kernel and device drivers for the specific mobile device;
2. Hypovisor contains the user space tools to setup and launch containers and virtual instances and supports the GNU tools that are found on a typical Linux distribution;
3. Isolated virtual instances cannot detect the hypovisor's presence, allowing the host instance to use additional tools to silently monitor the virtual instances;
4. SE Linux incorporated in the kernel helps in writing individual policies for the applications and makes sure that no applications acquire more information than needed;
5. Integrity auditing tools, e.g. Tripwire, help monitor the filesystem for changes, and unauthorized changes are informed to the user; and
6. Isolated instances of Multipersona do not communicate with each other, thus, there is no transfer of information between the instances.

The isolated namespaces are logical within the kernel, which uses to separate the processes. As long as the kernel and Angstrom's root user has not been compromised, a process that is within a child namespace will be jailed to that namespace

C. SE Linux Implementation

While originally based on a custom port of SE Linux to Multipersona, the ultimate implementation draws heavily from the SE Multipersona project. The key difference is that the SE Linux userspace tools are ported to the hypovisor instance of Angstrom, rather than as userspace tools for Multipersona. In fact, the Multipersona containers need not know SE Linux is present on the underlying system, as long as the kernel does the necessary filesystem markings and policy enforcement.

To develop policies for the hypovisor, SE Linux was run in permissive mode, where policy violations are logged. Using the audit2allow tool, appropriate policies for the hypovisor were formulated. Key variations from a typical embedded installation are the components necessary to support LXC containers, which leverage a variety of unique kernel features. Policies have been written for individual applications to prevent them from accessing more information than necessary. Setool was used for writing the policies. Policies were written in mac_permissions.xml file and multipersona source was recompiled.

The policies deployed for the subordinate Multipersona in-stances are minor variants of the stock SE Multipersona policies, with tweaks necessary to support some of the containerized hardware resources.

D. Memory Impact

Measurement of the both the Angstrom hypervisor running a container and native Multipersona were completed to understand the impact the hypovisor has on memory availability, which is important on a resource constrained device.

Utilities like “TOP” and “Free” were used to do the measurements. Free measures the virtual memory, allocated memory and free memory. TOP measures the same information and also gives information on how much memory is consumed by each process. Three tests were computed using a PandaBoard running three different scenarios. In the first scenario, a stock instance of Multipersona was running using a native root filesystem. In the second scenario, the Multipersona hypovisor running Angstrom Linux is booted, with no container instances active. In the final scenario, an instance of Multipersona is booted on top of the Hypovisor.

Table 1 presents the results, which depending on interpretation demonstrate that the overhead for operating

the hypovisor is approximately 40-50 MB of memory. The numbers are not strictly additive due to changes required to the kernel and Multipersona installation necessary to support Hypovisors. Also, additional processes such as the user-space container tools LXC are running within the Angstrom system. Lastly, the native Multipersona used (Linaro Multipersona) has additional memory optimization techniques that the containerized Multipersona does not utilize.

V. CONCLUSION AND FUTURE WORK

Key future work will focus on further memory optimizations, allowing multiple, simultaneous instances of Multipersona to share a common base of shared libraries. Key challenges will center on developing SE Linux policies to protect a common zygote operating across multiple security domains. Additionally, the hardware multiplexer shown in Figure 4 is rudimentary and needs further extension to support a variety of advanced features, such as 3D acceleration supported by the Cells project [2].

Our architecture integrates three major tools for improving Multipersona OS security: Linux containers, integrity tools, and a custom adaptation of SE Linux. Containers allow us to isolate instance(s) of Multipersona from an underlying hypovisor. That hypovisor both manages the multiple instances in a light-weight manner, and also provides a location for running a variety of integrity tools.

SE Linux promotes the notion of least privilege across the overall system and minimizes the probability with which malware can exploit containerized instances of Multipersona and compromise the underlying hypovisor. A reference implementation was developed on the PandaBoard running Multipersona 4.0 on Linux kernel 3.0.8. Tests showed that there was a minimal memory footprint for the hypovisor. The boot time is longer than a regular Multipersona device, but rest of processes like IO and startup time of applications is the same.

Overall, these tools remain a piece of a larger defense in depth security strategy for mobile devices. Application layer tools are still required to mitigate installation of malicious apps. Hardware-layer tools are necessary in certain environments to provide hardware roots of trust. Enterprise mobile device management and mobile app stores are necessary to better integrate mobile devices into an enterprise network and improve manageability and policy compliance.

ACKNOWLEDGMENT

Our thanks to Joe Tront, Ingrid Burbey, Michael Fowler, Randy Marchany, Stephen Groat, Dennis Kafura, Sonya Rowe, Philip Balister, Bob Lineberry, and Rick Cooper for their support and timely guidance. This was supported by the L-3 Communications National Security Solutions Center and the Naval Postgraduate School under contract N00244-11-P-2026.

REFERENCES

[1] The angstrom distribution. http://www.angstrom-distribution.org, October 2012.

[2] Andrus J. Dall, C. Hof A V, O. Laaden and J. Nieh Cells: “A virtual mobile multipersona architecture”. In ACM Symposium on Operating System Principles (SOSP) (October 2011), pp. 173-187.

[3] S. Bhattiprolu, E. Biederman , S. Hallyn, and D. Lezcano “Virtual servers and checkpoint/restart in mainstream linux”. ACM SIGOPS Operating Systems Review 42 (July 2008), pp. 104-113.

[4] S. Bugiel, L. Davi, A. Dmitrienko, T. Fisher, and A. Sadeghi, “XMmultipersona: A new multipersona evolution to mitigate privege escalation attacks”. Tech. Rep. TR-2011-04, Technische University Darmstadt, 2011.

[5] J. Burns “Mobile application security on multipersona”. Black Hat USA (2009).

[6] L. Davi, A. Dmitrienko, A. Sadeghi, and W. Winandy “Privilege escalation attacks on multipersona”. In Information Security Conference (ISC) (2010).

[7] W. Enck, P. Gilbert, B. Chun, L. Cox, J. Jung, P. McDaniel, and A. Sheth “Taintdroid: An information-flow tracking system for realtime privacy monitoring on multipersonas”. In USENIX Conference on Operating System Design and Implementation (OSDI), 2010.

[8] W. Enck, M. Ongtang, and P. McDaniel, “On multipersona mobile phone application certification”. In ACM Conference on Computer and Communications Security (CCS) (2009).

[9] C. Fleizach, M. Liljenstam, P. Johansson, G. Moelker, and A. Mehes. “Can you infect me now? Malware propagation in mobile phone networks”. In ACM Workshop on Rapid Malcode (WORM) 2007, pp. 61-68.

[10] D. Kafura, D. Gracanin , M. Perez-Quinones, and T. DeHart, “An approach to community-oriented email privacy”. In IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT) 2011, pp. 966- 973.

[11] G. Kim, and E. Spafford,”The design and implementation of tripwire: A file system integrity checker”. Computer Science Technical Reports 93-071, Purdue University, November 1993.

[12] Nils. “Building andriod sandcastles in multipersona's sandbox”. In Blackhat Abu Dhabi 2010.

[13] J. Oberheide “Multipersona hax”. In Proceedings of SummerCon 2010.

[14] M. Polychronakis, P. Mavrommatis, , and N. Provos, “Ghost turns zombie: Exploring the life cycle of web-based malware”. In USENIX Workshop on Large-Scale and Emergent Threats (LEET) 2008.

[15] P. Ruggiero, and J. Foot, “Cyber threats to mobile devices”. Tech. Rep. TIP-10-105-01, United States Computer Emergency Readiness Team, April 2010.

[16] R Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang “Soundcomber: A stealthy and context-aware sound trojan for multipersonas”. In Network and Distributed System Security Conference (NDSS) 2011.

[17] A. Shabtai “Securing multipersona-powered mobile devices using SE Linux”. IEEE Security and Privacy 8, 3 May 2010, pp. 36-44.

[19] D. Shetty “Demystifying the multipersona malware”. SecurityXploded: An Infosec Research and Development Portal, September 2011.

[20] D. Sin, J. Ahn, and C. Shim, “Progressive multi gray-leveling: A voice spam protection algorithm”. IEEE Network 20, 5, September 2006, pp. 18- 24.

[21] S. Smalley, “The case for SE Multipersona”. In Linux Security Summit, 2011.