



TECHNOLOGY WHITE PAPER

Android Migration at the Speed of Light

How virtualization can reduce time, risk, and cost when moving to the Android platform

Josh Matthews
Field Application Engineer
Open Kernel Labs, Inc.

June 9, 2009

Growing success and looming challenges

Android shows every sign of taking the mobile handset market by storm. Google recently forecasted that up to 20 Android phones would be available before the end of this year. This is amazing progress for a platform first announced only 18 months ago. The benefits of the environment are evident. It's not just an OS, but also a complete software development environment for mobile devices – and all for free. Android's very success gives it momentum with rapid acceptance, and an open source environment driving a fast-growing ecosystem of application developers. So successful, in fact, that even Microsoft has identified it as a serious threat to their future.

However, the very success of Android presents its own set of challenges. The first issue is simply the size of the task – Android is a complete handset platform, combining a mobile operating system, GSM stack, and middleware, plus browser and application environment. The implementation of such a comprehensive package in a handset design would appear to demand an all-or-nothing approach to the development task; existing investments, innovations, and expertise built on current platforms must be abandoned in favor of an all-encompassing new technology. This would be problem enough, but the time-to-market pressures in this rapidly evolving market make the developer's life even more difficult.

Additionally, although platform standardization has its benefits, it presents a corresponding problem – how can vendors create a competitive edge for their product? Indeed, much of the differentiation and branding of existing devices is locked up in the mobile OEM/mobile network operators' (MNOs) proprietary legacy code. Unfortunately, this must all be ported to the new Android environment.

Legacy software may also demand porting to a new processor architecture, because of the higher-end hardware demanded by this more capable, and hence more demanding, mobile platform. Indeed, current Android designs are all dual-core with high-powered, dedicated application processors such as ARM11. Beyond the additional effort required to port to a new processor architecture, the resulting higher cost significantly restricts the ability of the mobile OEM to address the needs of the large segment at the lower end of the market.

Android migration in the fast lane

These challenges are daunting. But what if there were a way to greatly reduce the risk and time-to-market overhead associated with the migration task? And what if this could be done with increased flexibility in choice of target hardware design and product features to promote differentiation? With the enhanced mobile phone virtualization environment provided by the OKL4 microvisor, all this is possible.

Using the mobile phone virtualization approach instead of sacrificing legacy investments, device manufacturers and MNOs can migrate to Android with little or no migration cost for legacy code, such as existing application stacks. This allows key differentiating features of existing devices to be retained, while simultaneously reducing risk and development time.

This is not the only benefit of this approach. The typical hardware platform for an Android handset is a dual processor design, usually ARM9 + ARM11. A unique capability of a virtualized environment is that both single- and dual-processor designs are perfectly possible, giving the manufacturer a much greater choice of target silicon for a handset design and much more flexibility on price point.

The OKL4 microvisor has already achieved commercial success with such a virtualized single-core design: the recently launched Motorola Evoke QA4 device, heralded as the “best touch-screen from Motorola,” seamlessly integrating a Linux application environment, baseband stack, and several legacy applications, on a single-core ARM9.

There is yet another advantage of virtualization when migrating to Android – once the design is virtualized, any subsequent port to a different hardware architecture – e.g. single- to dual-core, or even from dual- to single-core – is greatly simplified.

These are some big claims. To back them up, we'll look at the OKL4 microvisor, show how Android works in a virtualized environment, explain how this simplifies the migration task, and describe some specific examples of its application.

Mobile phone virtualization done right

The OK Labs microvisor combines operating system, embedded hypervisor, virtualization, and componentization capabilities in a very small piece of system software. At its core is the OKL4 microkernel, the commercially-distributed and commercially-supported member of the L4 microkernel family. The OK Labs microvisor environment is based on 14 years of research leadership in microkernel technology and is solidified by several years of successful commercial engagements and over 300 million deployments in mobile devices.

The key capability of the microvisor is its Secure HyperCell™ Technology¹. This unique architecture provides a development, deployment, and execution engine for both coarse-grained virtual machines and fine-grained embedded components, any of which may be securely isolated in individual cells. The microvisor has complete and sole control of the underlying hardware and is the only software running in privileged mode; each microvisor cell partitions and multiplexes this hardware between any other software on the target system as required, from high level OSes down to individual applications and drivers. Figure 1 shows a typical example of such a system.

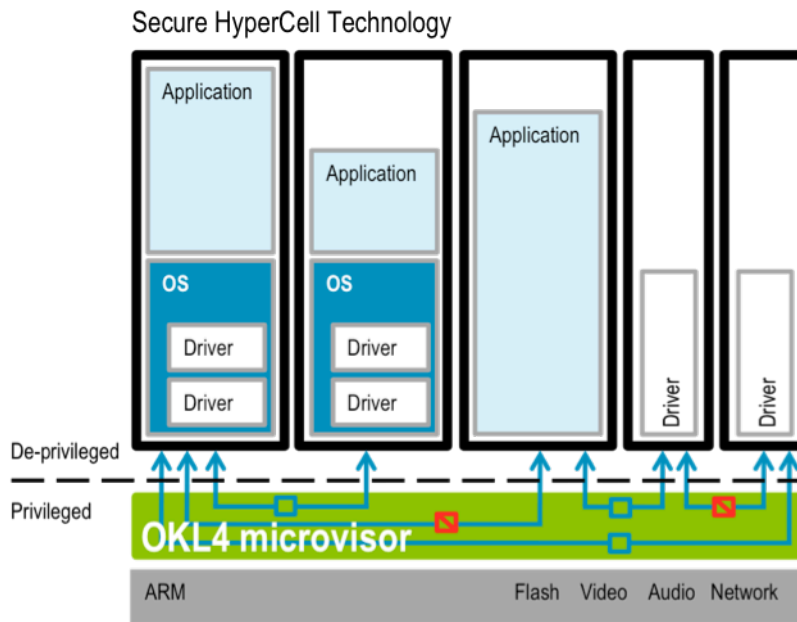


Figure 1: OKL4 microvisor with Secure HyperCell Technology

Here we can see the unique capability of the OKL4 microvisor - and this is the key feature in enabling fast migration to Android - a cell may contain components at a much finer-grained level than is the case for conventional virtualization environments. This means that any legacy subsystems can be isolated from the Android environment and executed directly on the microvisor.

Migration with the microvisor

The introduction of the OKL4 microvisor to an existing engineering process is extremely low-impact; the goal is to provide the greatest flexibility to system designers by enabling any new or legacy subsystem to be treated as an isolated component. This allows system engineers to continue to work with any legacy component they wish to migrate, just as they always did, in its native environment. In keeping with this approach, the microvisor environment is structured to provide a simple, four-step process to a complete, rapid, and risk-reduced Android migration path:

1. **Integration of OKL4 with the target hardware.** The OKL4 microvisor supports all common embedded mobile processors; the system designer simply integrates the microvisor with the particular combination of devices present on their hardware, typically an integrated SoC design;
2. **Mobile phone virtualization and integration of Android.** The microvisor already has an off-the-shelf pre-virtualized OK:Android implementation available. The system designer receives this as a distributed secure cell component and integrates into their final system using OKL4 microvisor tools;
3. **Mobile Phone Virtualization and introduction of the communications stack.** A communications stack component, most commonly running on an RTOS, can be

hosted without change in the OKL4 microvisor environment. This alone is a major advantage in time-to-market, as it avoids the need to recertify the code for the network operator:

- 4. Migration of legacy components.** Selected legacy components are migrated to the OKL4 microvisor in a straightforward manner using OKL4 compatibility libraries, and integrated into the final system as a cell. One obvious candidate here would be a custom GUI/application stack, which can be ported unchanged, saving time and also preserving product differentiation features.

Now let's look at each of these steps in more detail.

Porting the microvisor to an SoC

With many years of experience of this process, it's no surprise that OK Labs has all of the tools in place to support a port to an SoC design for mobile applications. OK Labs offers a complete System on Chip Software Development Kit (SoC SDK) providing a simple and straightforward method for system engineers to integrate the OK Labs microvisor with their particular SoC design. The SoC SDK completely abstracts the SoC-specific implementation aspects of the microvisor from the OKL4 kernel itself.

By providing a set of SoC-specific stub functions, the SoC SDK serves as a guide for the SoC developer to implement those required specifically for their system configuration. Such stub functions cover all aspects of SoC functionality, including module startup, interrupt configuration and control, cache operations, timer operations, and SoC-specific debugging and error handling.

The output from this process is a reusable SoC module. The OKL4 system configuration tool is then used to combine this module with the OKL4 kernel provided, along with any system component cells, into the final bootable system image. A simple diagram of this process is shown in figure 2.

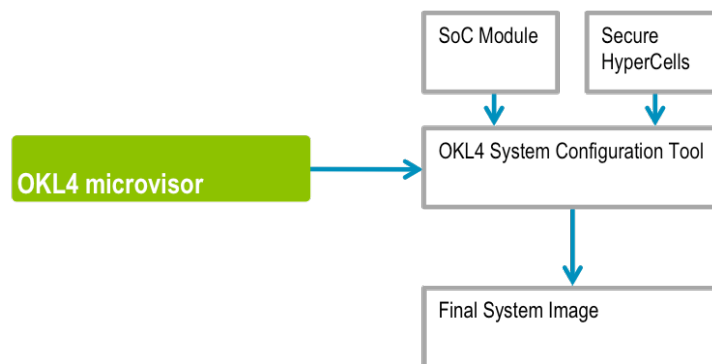


Figure 2: Integrating the OKL4 microvisor with an SoC design

Integration of OK:Android

OK:Android is a virtualized Android component, distributed and integrated as a secure cell under the control of the microvisor. System designers simply receive OK:Android and integrate it into their system utilizing the OKL4 system configuration tool. It is worth noting that other high-level operating systems, such as OK:Symbian are available for integration within the same system, allowing legacy applications running under other OSEs to co-exist in native form alongside Android if required.

Internally, OK:Android is engineered by OK Labs using a highly efficient approach known as *minimally invasive paravirtualization*. This involves no modification to the architecture-independent aspect of Android’s Linux kernel; the tiny (c. 8 kLOC) OKL4 target is introduced as a new architecture into the Linux *arch* directory. All Linux applications, including the rest of the Android platform stack, are completely binary compatible and execute unmodified.

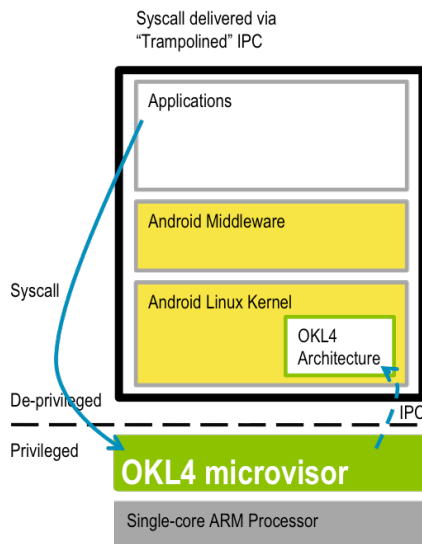


Figure 3: Android paravirtualization

As the OKL4 microvisor is the only software executing in privileged mode, when a *syscall* is executed by an Android application, the microvisor simply encodes the *syscall* into a message transmitted by its highly-efficient Inter-Process Communication (IPC) mechanism and quickly “trampolines” it to the Linux kernel executing in user mode. The OKL4 module only needs to decode the IPC message – after that, it just executes as standard Linux. See figure 3.

Virtualizing the communications stack

In a similar manner to the virtualization process we have just described, a communications stack (which typically executes on a real time operating system) can

also be virtualized. A number of common communications stacks for mobile devices are already virtualized for the OKL4 microvisor.

For those that aren't, OK Labs can provide simple but powerful services that allow for the swift implementation of paravirtualization for any RTOS. This approach does not just support the existing legacy codebase, but also means that the same legacy API is available for use by the application stack. This design approach is shown in figure 4.

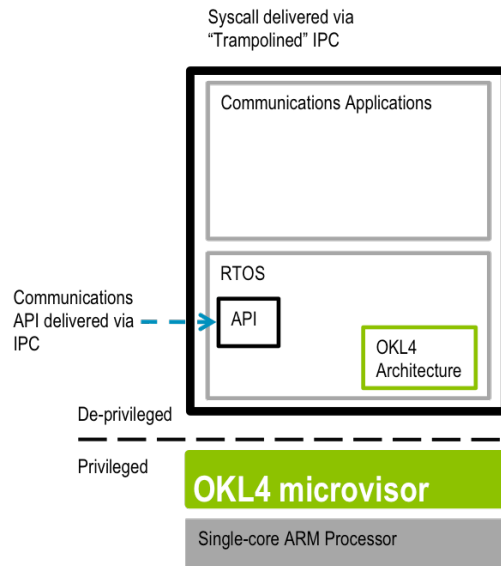


Figure 4: Communications stack paravirtualization

It is important to note that this process is highly efficient and fully capable of supporting the real-time performance demands of the virtualized RTOS.

Migration of legacy components

As described, one or more secure cells can each contain a single, strongly isolated, yet highly integrated, legacy component that executes directly on the OKL4 microvisor. As has been mentioned previously, these components can be of any size and type – from a high-level OS down to the simplest application routine. This allows any legacy code to operate in its own virtualized space, effectively independent of, yet closely coupled to (via the microvisor's IPC) the Android environment. See figure 5.

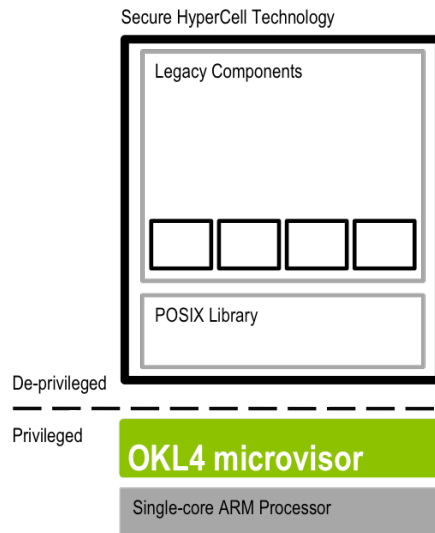


Figure 5: Migration and security for legacy components

To achieve this, the system integrator ports whichever legacy application or subsystem is required to the microvisor. This process is made straightforward by the inclusion of several compatibility libraries (including POSIX), which connect industry-standard functionality to OKL4 APIs.

A further benefit of the microvisor approach is the level of hardware independence offered by this virtualized environment. Effectively, the microvisor provides a “portability layer” that abstracts away from hardware issues such as the number of processors. This allows the developer to smoothly and rapidly move down to one processor or up to a dual-core design at any time, according to the demands of the application environment and cost constraints of their Bill of Materials costs (BOM). Rapidly moving from ARM9 to ARM11 or a Cortex A8/A9 becomes as straightforward as plugging in the Android OS stack and the existing pre-ported microvisor paravirtualization.

And for security-sensitive applications, the highly secure architecture provided by Secure HyperCell Technology automatically provides complete isolation for sensitive software components requiring the highest level of protection. Running these applications directly on the microvisor has a couple of specific advantages. The first is that they enjoy a much smaller trusted computing base than if they ran on a full Linux stack that powers the standard Android OS. The second is by controlling the IPC mechanisms connected to sensitive software, the entire software stack becomes more secure and less likely to experience a security breach.

Putting it all together

The final result of this straightforward, four-step process is the creation of an integrated design that combines Android, a communications stack, and any required legacy components on the target hardware, as shown in figure 6.

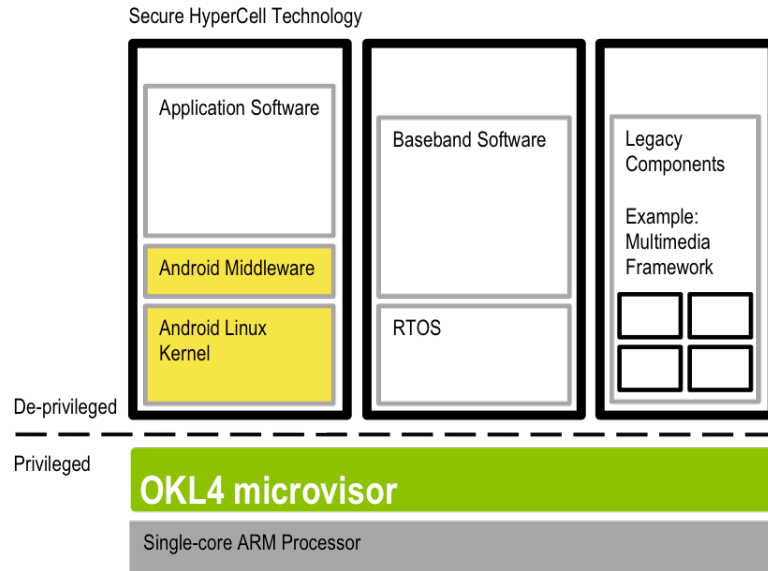


Figure 6: Fully virtualized Android architecture

The result is a highly flexible, highly secure environment that protects legacy code investment, both now and in the future. From our long experience at OK Labs, we know that porting to new hardware with the OKL4 microvisor is simpler and faster than with larger and more complex software environments, such as Linux, for many reasons. The key advantage is the fact that the microvisor is an extremely small software product with a small set of powerful services. When compared with larger software stacks such as Linux, which may require multiple engineers working for months to support a new platform, we have measured the porting effort to new platforms for comparable applications in a matter of days.

Typically, as embedded systems products are updated over time, development teams choose to migrate to a newer version of their chosen device operating system. In some cases the operating system may change completely from one generation of the product to the next. For example, the adoption of Linux for use in embedded systems has resulted in many products migrating from a proprietary embedded operating system to Linux with a new version of a product.

When either scenario occurs, a method must be found to reuse significant amounts of software that have already been developed, debugged, and validated with an older OS version or different OS. One approach is to port that software to the operating system chosen for the new project. The effort required to port and retest existing software with a new operating system reduces the benefit of software reuse.

The unique system virtualization capability provided by the OKL4 microvisor enables legacy software of all kinds to be reused in its original operating system environment, while new software is developed for a new operating system. By providing a secure cell for each required operating system or version, the microvisor environment reduces the

effort required to integrate legacy software with new software, allowing the creation of a new product release in less time and at lower cost.

This environment gives a developer the freedom to use any or all Android components freely in combination with other non-Android products and/or operating environments. An example of such a platform is one that combines Windows Mobile 7 or Symbian environment and applications with OK:Android. Such a mobile phone design would be most challenging if not impossible without mobile phone virtualization.

The need for speed

There is an understandable concern that all of these many benefits come at a price – namely, a performance penalty resulting from the virtualized environment. Perhaps surprisingly, in many applications – particularly resource-constrained applications such as a single-processor design – there is little or no loss of performance compared to a native implementation. The table shows some numbers from a real application using both native and virtualized environments.

Benchmark	Native (mS)	Virtualized (mS)	Overhead
GtkEntry	0.04	0.04	0
GtkComboBox	18.03	18.6	3%
GtkComboBoxEntry	14.21	14.57	2%
GtkSpinButton	2.57	2.64	3%
GtkProgressBar	1.15	1.16	1%
GtkToggleButton	4.08	4.23	4%
GtkCheckButton	4.19	4.26	2%
GtkRadioButton	7.74	7.85	1%
GtkTextView - Add text	11.69	11.71	0
GtkTextView - Scroll	6.63	6.35	-4%
GtkDrawingArea - Lines	12.25	11.86	-3%
GtkDrawingArea - Circles	25.64	25.04	-2%
GtkDrawingArea - Text	28.12	27.97	-1%
GtkDrawingArea - Pixbufs	4.24	4.2	-1%
TOTAL	140.58	140.48	0

The table shows performance figures (in microseconds) of GtkPerf benchmarks executing on an ARM926ejS with 126MB RAM, running at 240MHz. The environment was Linux 2.6.24 and OKL4 3.0.1. What is perhaps surprising is that in some cases, the virtualized code actually executes faster! Overall, in this case at least, the outcome was close to a dead heat. On reflection however, we should not be too surprised.

This is a good illustration of a major goal for OK Labs – to achieve zero-overhead paravirtualization for high-level operating systems such as Linux. Linux was never really intended or designed for efficient operation in real-time embedded applications. There are often several optimizations available for particular OS/processor combinations that make a speed gain possible when the OS is paravirtualized, which in many cases will eliminate any performance penalties due to the virtualized environment.

It may be interesting to examine some specific examples where the implementation of the OKL4 microvisor and OK:Android provides enhancements to system performance. Examples will be taken primarily from our ARM9 experience, which as a lower-cost architecture, extracting the maximum performance is a major concern. In particular, we will show how the OK Labs mobile phone virtualization environment can reduce memory footprint and improve performance on such low-end devices that enables low-cost phones to compare favorably in performance and user experience with more expensive designs.

Fast address space switching

One of the most important benefits to system performance provided by the OKL4 microvisor is the ability to perform an extremely fast, low-latency context switch between different processes.

The Android software architecture is built on isolated components communicating via a custom IPC mechanism. This component architecture means that there is more communication between processes than is usual in a traditional Linux-based operating environment. In Linux, a full-cache flush is required on each context switch between processes, leading to additional latency and memory bus usage. This allows significant optimizations to be implemented, particularly in lower performance architectures such as the ARM9, commonly used in many cost-sensitive mobile designs.

When switching between different components in the microvisor environment, the cache state is maintained, leading to both improved context switch latency and better overall performance through improved cache utilization. As another example of memory-efficient implementation, the standard ARM page-table is based on a format that uses a 16-kilobyte page directory for every process. OKL4 is able to provide a special compressed page-table format that can significantly reduce the amount of memory required for storing a process's virtual address space translations.

High-performance shared libraries

In addition to reducing page-table size of individual processes through compressed page-tables, OKL4 has the potential to lower the memory usage of its high-performance shared libraries. Rather than shared libraries consuming page-table memory in each individual process, a single set of page-tables can be used for all processes in the system, and shared using ARM's domain mechanism.

Not only does this represent significant savings in page-table memory, it will also enable better cache utilization, as shared libraries will be shared in the cache. In an architecture where all user applications are very similar (i.e. all running the same virtual machine), this additional benefit of sharing cache memory will have a major impact on performance.

Low footprint memory usage

The OKL4 microvisor is optimized for performance and memory usage. It is difficult to directly compare the memory size of a monolithic kernel such as Linux, and a microkernel-based environment such as the microvisor, as Linux obviously provides a different range of services from the kernel. However, it is worth noting that the OKL4 microkernel itself is measured in just a few tens of kilobytes, while the Android Linux kernel is measured in megabytes. The inherently small size of the underlying kernel does not, by itself, mean that the combined system will have a small footprint, but when a small base is combined with careful selection of additional components, the resulting paravirtualized OK:Android can be relatively lightweight.

Just like Android – but better, faster, cheaper!

Smartphones are rapidly getting smarter, with a massively increasing range of features and capabilities. Time-to-market pressures are building. Android 2.0 is just around the corner. The challenges faced by the handset developer, as a result of the complexity of the development process and of the business model, are profound.

But now there is a solution. By decoupling the software environment from the underlying hardware, OK Labs puts the developer back in control of this process, both at the engineering level and also at the commercial level. The OKL4 microvisor/OK:Android virtualized environment provides the highest level of flexibility in terms of hardware and software choices, ensures protection of legacy code investment, maintains product differentiation, and delivers the fastest time-to-market with the lowest development risk.

And when it comes time to port the virtualized environment to a higher performance hardware platform – or even move down to a lower cost handset BOM - the future-proof design will port with the very minimum of effort. This is just what Android was meant to be – better, faster & cheaper!

References

Ref. 1: Secure HyperCell Technology

<http://www.ok-labs.com/products/product-strategy/secure-hypercell-technology>

About the author

Josh Matthews, lead Field Application Engineer for Open Kernel Labs, works closely with customers to identify solution requirements and manage successful deployments of mobile virtualization. His experience spans the mobile ecosystem, including work with handset OEMs, semiconductor suppliers, and MNOs across the globe from Asia to Europe and the US.

Matthews' work on Android is helping engineers design more high-functioning devices at a lower cost, extending significant savings through out the mobile ecosystem.

Matthews has extensive experience in embedded hypervisor and microkernel technology, studying under Dr. Gernot Heiser and working as a kernel hacker with the OK Labs engineering team in Sydney. He is a frequent presenter at conferences and seminars and author of numerous technical articles and white papers. He is now based in the US.

About Open Kernel Labs

Open Kernel Labs is the global leader in open source virtualization software for mobile devices, consumer electronics, and embedded systems. Backed by the largest, independent team of microkernel developers, the OKL4 microvisor is deployed in more than 300 million mobile phones worldwide. Semiconductor suppliers, handset OEMs, and mobile network operators depend on OK Labs to deliver high performance solutions that decrease BOM cost, reduce complexity and speed time-to-market.

For information on the OK Community, please visit the Community Portal at www.ok-labs.com/community/community-portal. Participants can join the Developer's Mailing List at <http://www.ok-labs.com/community/mailling-list-signup>.

Open Kernel Labs, OK Labs and Secure HyperCell™ are trademarks or registered trademarks of Open Kernel Labs or its affiliates in the U.S. and other countries. Other names may be trademarks of their respective owners. All other trademarks and registered trademarks are property of their respective owners.